MySQL Shell 8.0 Release Notes

Abstract

This document contains release notes for the changes in each release of MySQL Shell 8.0.

For additional MySQL Shell documentation, see http://dev.mysql.com/.

Updates to these notes occur as new product features are added, so that everybody can follow the development process. If a recent version is listed here that you cannot find on the download page (https://dev.mysql.com/downloads/), the version has not yet been released.

The documentation included in source and binary distributions may not be fully up to date with respect to release note entries because integration of the documentation occurs at release build time. For the most up-to-date release notes, please refer to the online documentation instead.

For legal information, see the Legal Notices.

For help with using MySQL, please visit the MySQL Forums, where you can discuss your issues with other MySQL users.

Document generated on: 2021-01-29 (revision: 21846)

Table of Contents

Preface and Legal Notices

This document contains release notes for the changes in each release of MySQL Shell 8.0.

Legal Notices

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit
https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Changes in MySQL Shell 8.0.24 (Not yet released)

Version 8.0.24 has no release notes, or they have not been published because the product version has not been released.

Changes in MySQL Shell 8.0.23 (2021-01-18, General Availability)

AdminAPI Added or Changed Functionality

- The output of the `status()` operation has been extended to provide more information relevant to diagnosing errors. The following information is available for InnoDB Clusters and InnoDB ReplicaSets:
  - the `memberState` field shows the actual status of the instance as queried locally, which can be one of `offline`, `error`, `recovering`, or `online`.
  - a `recovery.recoveryChannel` field shows instances performing incremental recovery or in which the recovery channel status is not off
  - a new `instanceErrors` field exists for each instance, displaying any diagnostic information that can be detected for it
  - when the `extended` option is set to greater than 0, the output includes an `applierChannel` field, with replication information if the instance is either online and the applier channel status is not on, or the status is not recovering or online and the applier channel status is not off

  For more information, see Checking a cluster's Status with *`Cluster`*`.status()`.

- InnoDB Cluster and InnoDB ReplicaSet now support and enable parallel replication appliers, sometimes referred to as a multi-threaded replica. With the advances in MySQL such as binary log transaction dependency tracking and XXHASH64 based GTID set extraction, using multiple replica applier threads improves the throughput of both the replication applier and incremental recovery.

  This has resulted in the following changes:

  - the requirements for instances running 8.0.23 and later now also include:

    - `binlog_transaction_dependency_tracking=WRITESET`
    - `slave_preserve_commit_order=ON`
    - `slave_parallel_type=LOGICAL_CLOCK`
    - `transaction_write_set_extraction=XXHASH64`

    this means that new instances running 8.0.23 have these options configured by `dba.configureInstance()` and `dba.configureReplicaSetInstance()`. Attempting to add an instance running version 8.0.23 or later which does not have these variables configured results in an error. When you upgrade a cluster or replica set that has been running a version of MySQL server and MySQL Shell earlier than 8.0.23, the parallel replication applier is not enabled on the instances. This means you are not taking advantage of this feature, and you should reconfigure your instances to use the parallel replication applier. For more information, see Configuring the Parallel Replication Applier.

  - `dba.checkInstanceConfiguration()` validates if parallel replication appliers are enabled or not.
  - the new `applierWorkerThreads` option configures the number of replication applier threads the instance uses for replication, and defaults to 4 threads. Use this option with the `dba.configureInstance()` and `dba.configureReplicaSetInstance()`. You can change this option while the instance is online, but the change is only made after the instance is restarted.
  - the output of the `.status(extended=1)` and `options()` operations now includes information about the configuration of parallel appliers.

AdminAPI Bugs Fixed

- The fix for Bug#29305551 extended the `dba.checkInstanceConfiguration()` operation to include a check to verify if asynchronous replication is configured and running on the target instance, and print a warning when that is the case. This check is also used by the *`Cluster`*`.addInstance()` and *`Cluster`*`.rejoinInstance()` operations to terminate them with an error when such a scenario is detected, and is also used by the `dba.rebootClusterFromCompleteOutage()` operation whenever there are instances to be rejoined to the cluster. However, the `dba.createCluster()` operation was erroneously skipping the check, and the `dba.rebootClusterFromCompleteOutage()` operation was skipping the check on the instance being used to bootstrap the cluster. The fix ensures that the check is also performed whenever creating or rebooting a cluster from complete outage. Additionally, it adds support to override the check for the `dba.createCluster()` operation by making use of the `force` option, and it improves the error messages. (Bug #32197222)
- The fix for Bug#29305551 extended `dba.checkInstanceConfiguration()` to verify if asynchronous replication is configured and running on the target instance and print a warning if that was the case. However, the check missed verifying if the replication channel was configured but not running. This fix ensures the verification also considers replication channels which are configured but are not actively running. Additionally, an erroneous message which suggested the possibility of using `STOP REPLICA` to override this check has been removed and replaced with an informative message

which explains that unmanaged replication channels are not supported and the possible dangers of their usage. (Bug #32197197)

- Based on the terminology changes in WL#14189, AdminAPI has been aligned with the new terms. Error and log messages now use the terms source (previously master) and replica (previously slave). (Bug #32152133)

- During a *`Cluster`*`.rebootClusterFromCompleteOutage()` operation, the GTID superset is used to detect which instance should be used to reboot the cluster. If an instance had a diverging GTID set and you wanted to explicitly remove it from the cluster, the operation blocked because it could not determine which instance had the GTID superset. Previously, in such a situation there was no way to exclude the instance from the instances used to detect the GTID superset. Now, if you answer no during the interactive wizard, or configure the `removeInstances` option, the instance is not checked as part of finding the GTID superset. (Bug #32112864)

- When an instance had left a ReplicaSet, and then its configuration was changed in a way that made it invalid for InnoDB ReplicaSet usage, the *`ReplicaSet`*`.rejoinInstance()` operation did not detect that the configuration was invalid. Now, instances are checked to ensure they are valid before rejoining them to a ReplicaSet. (Bug #31975416)

- When upgrading the metadata using `dba.upgradeMetadata()`, if there are MySQL Router instances that need to be upgraded, the operation waits until all instances are upgraded before continuing. The operation offers you an option to re-check for outdated MySQL Router instances and continue with the metadata upgrade. A MySQL Router upgrade is only complete after a restart of the application, however the message printed did not mention that. This message now includes the information that MySQL Router instances must be restarted after the binaries are upgraded. (Bug #31882876)

- When you were connected to a secondary instance, attempting to issue operations such as *`Cluster`*`.rejoinInstance()`, *`Cluster`*`.addInstance()`, *`Cluster`*`.dissolve()` and so on would fail. Now, AdminAPI always connects to the current primary.

  As part of this work the following changes were made:

  - Now, in the event that `dba.createCluster()` or *`Cluster`*`.addInstance()` fail with a Group Replication error, AdminAPI returns the `performance_schema.error_log` entries.

  - The *`Cluster`*`.rejoinInstance()` operation has been changed to succeed if the instance is already in the cluster, instead of throwing an exception.

  - The `dba.rebootCluster()` operation has been changed to not clear `super_read_only` on the instance.

  (Bug #31757737)

- As part of the default settings for InnoDB Cluster, to ensure that instances automatically rejoin the cluster, the `group_replication_start_on_boot` option is automatically set to true. However, this meant that in environments with an external tool managing the cluster life cycle, for example an orchestrator such as Kubernetes, the automatic enabling of rejoin could cause conflicts with the tool. In addition, if the automatic rejoining of an instance was enabled at an unsuitable time (for example when rebooting, or while repairing a split-brain, and so on), a deadlock or long freezes could occur until a timeout happened. In some situations, instances could even potentially join the wrong cluster during a reconfiguration.

  To avoid such situations, the `manualStartOnBoot` boolean option has been added, which defaults to false. To disable the automatic rejoining of an instance, for example while repairing a split-brain, set the `manualStartOnBoot` option to true. This prevents the instance rejoining the cluster automatically while you make changes. You then need to rejoin the instance to the cluster manually, before setting

the `manualStartOnBoot` option back to false to ensure instance it rejoins the cluster automatically again. Similarly, if you are using an external orchestrator to manage the life cycle of instances, set the `manualStartOnBoot` option to true across the whole cluster, to disable the automatic rejoining of instances to the cluster. Your orchestrator should then be configured to rejoin the instances manually. (Bug #31643595)

- Calling `dba.checkInstanceConfiguration()` with `verifyMyCnf` set to a file which did not exist, the operation completed successfully saying the configuration file had been checked. The fix checks if the file specified by `verifyMyCnf` exists, prints an error if not, and ensures the console does not show unnecessary error messages. (Bug #31468546)

- On an instance with the `sql_mode` variable set to `ANSI_QUOTES`, attempting to upgrade the metadata schema with `dba.upgradeMetadata()` failed with the error: `Unknown column 'MySQL Router' in 'field list'`. This was related to a query which uses single quotes to quote strings. As part of this fix, the upgrade metadata operation now prepares the session to be used by AdminAPI, and amongst other sanity checks it ensures that the `sql_mode` for that session uses the default value to avoid incompatible user configured settings. Additionally, the same was done for the `dba.getCluster()` and `dba.dropMetadataSchema()` operations. (Bug #31428813)

- If the MySQL Shell global session was connected to a sandbox instance, and that instance was stopped, MySQL Shell tried to incorrectly reconnect to the instance. Now, if the active session is connected to a sandbox instance which is being stopped, MySQL Shell closes the session. (Bug #31113914)

- The output of *`Cluster`*`.status()` now includes additional information about instances that are registered in the metadata but not currently online. MySQL Shell now connects to offline instances found in the metadata and attempts to diagnose them, providing additional information such as their connectivity and status. (Bug #30501615)

- Instances that are part of the underlying group but are not identified in the metadata, for example because they were configured manually and bypassing MySQL Shell, or because they were previously removed from the InnoDB Cluster but were not properly decommissioned, are now shown in the output of *`Cluster`*`.status()`, along with diagnostic warnings about the metadata discrepancy. This ensures you can detect situations where an instance is participating in the group but is not being managed by MySQL Shell. (Bug #27882663)

- An instance that belongs to an InnoDB Cluster is identified by its server UUID. If the UUID changed after the instance had left the cluster, for example because you used MySQL Enterprise Backup to restore from a backup, then the instance could not be rejoined to the cluster. Now, if the cluster encounters this situation, it checks the metadata to see if the instance can be identified using its host and port. If found, the metadata is updated based on the options used for the rejoin operation. This check is executed during the *`Cluster`*`.rejoinInstance()` and *`Cluster`*`.rescan()` operations.

  Additionally, a check is executed to verify the `serverId` of all the instances is registered in the metadata as an instance attribute. If it is not, the metadata is updated accordingly. This check is executed on add, rejoin and rescan operations. (Bug #26649039)

Functionality Added or Changed

- MySQL Shell's parallel table import utility can now import a specified list of input data files, and it supports wildcard pattern matching to include all relevant files from a location. Multiple files uploaded by a single run of the utility are placed into a single relational table, so for example, data that has been exported from multiple hosts and stored in multiple files could be merged into a single table to be used for analytics. The files can be compressed in the gzip or zstd format, and in that case the utility reads them from storage in the compressed format, saving bandwidth for that part of the transfer. The utility then uses its parallel connections to decompress and upload several files simultaneously to the target server.

Bugs Fixed

- When MySQL Shell's instance dump utility `util.dumpInstance()` was run with the `ocimds` option set to `true` to check compatibility with MySQL Database Service, and the `users` option set to `true` to include users and their roles and grants in the dump, the utility reported some compatibility errors for privileges that actually were permitted. MySQL Shell's allowed list of privileges for MySQL Database Service has now been updated. (Bug #32213605)

- The behavior of MySQL Shell's table dump utility `util.dumpTables()` and dump loading utility `util.loadDump()` regarding the schemas for single table dumps and loads has been changed. Previously, the dump files produced for a single table did not contain the SQL statements to recreate the schema, so the schema had to exist in the target MySQL instance before the dump loading utility could load the table. Now, the dumps produced by the table dump utility contain the schema statements, and when they are loaded with the dump loading utility, by default, the schema is created in the target MySQL instance if it does not already exist. The `schema` option can be used to load the table dump into another schema that exists in the target MySQL instance. Table dumps created using the earlier version of the utility still require the `schema` option and an existing schema. (Bug #32165101)

- MySQL Shell's table dump utility `util.dumpTables()` now supports the `ocimds`, `compatibility`, `ociParManifest`, and `ociParExpireTime` options, so you can check compatibility with MySQL Database Service, and generate pre-authenticated request URLs for the dump files. Also, the `ignoreVersion` option has been extended to allow the import of a dump that was created without the `ocimds` option into a MySQL DB System. (Bug #32140970)

- If a dump included users that were created with external authentication plugins, MySQL Shell's dump loading utility `util.loadDump()` was unable to load the dump if those plugins were not available on the target server instance. The `ocimds` option for MySQL Shell's instance dump utility `util.dumpInstance()` and schema dump utility `util.dumpSchemas` which checks compatibility with MySQL Database Service, now checks for accounts using authentication plugins that are not supported in MySQL Database Service. The compatibility option has an additional modification option `skip_invalid_accounts`, which removes such user accounts. (Bug #32115948)

- Previously, MySQL Shell's dump loading utility `util.loadDump()` stopped with an error if the `loadUsers` option was set to true but the supplied dump files did not contain user accounts. The utility now displays a warning and continues in this situation. (Bug #32115861)

- MySQL Shell's instance dump utility `util.dumpInstance()`, schema dump utility `util.dumpSchemas()`, and table dump utility `util.dumpTables()` falls back to using the `LOCK TABLES` privilege to lock dumped tables if the consistent option is set to true, which is the default, and the `RELOAD` privilege is not available. However, the locking operation could cause an implicit commit on active transactions, meaning that the data was not dumped consistently. The locking has now been corrected to ensure consistency in this situation. (Bug #32107327, Bug #101410)

- When MySQL Shell's dump loading utility `util.loadDump()` used indexes to identify row boundaries, an error occurred if an index pointed beyond the data in the read buffer. The utility now checks for this situation and ignores the index if so. (Bug #32072961)

- When MySQL Shell was attempting to reconnect to a server, **Ctrl + C** did not interrupt the operation. The interrupt now functions and sets the retry attempts counter to zero so that the sequence exits correctly. (Bug #32041342)

- MySQL Shell can now be built using Python 3.9. (Bug #32020230)

- The `updateGtidSet` option for MySQL Shell's dump loading utility `util.loadDump()` could not be used with MySQL DB System due to a permissions restriction. The utility now uses a stored procedure that is permitted, so the option can be used. (Bug #32009225)

- When MySQL Shell's instance dump utility `util.dumpInstance()`, schema dump utility `util.dumpSchemas()`, or table dump utility `util.dumpTables()` was exporting to an Oracle Cloud Infrastructure Object Storage bucket, if there was a loss of connectivity or routing to the Object Storage server, MySQL Shell stopped unexpectedly. The error is now handled correctly. (Bug #32005418)
- MySQL Shell's dump loading utility `util.loadDump()` returned an exception if a header value in a response was empty. (Bug #31979374)
- MySQL Shell did not initialize Python 3.8's new `cf_feature_version` compiler flag field, which could cause an exception when format strings were used. (Bug #31926697)
- Where MySQL Shell is using a system installation of Python rather than the bundled version, the minimum version that MySQL Shell supports is now Python 3.6. Python 3.4.3 was the previous minimum for a system installation. The bundled version is Python 3.7.7. (Bug #31900744)
- MySQL Shell's instance dump utility `util.dumpInstance()`, schema dump utility `util.dumpSchemas()`, and table dump utility `util.dumpTables()` use table statistics to identify a suitable default row size. If the statistics for a table are outdated or not present, this can cause issues for the chunking process. In this situation, MySQL Shell now issues a message to suggest using an `ANALYZE TABLE` statement to produce up to date statistics. (Bug #31766490)
- The `skipBinlog` option for MySQL Shell's dump loading utility `util.loadDump()` skips binary logging on the target MySQL instance for the import. The option is not suitable for MySQL DB System as the binary logging status cannot be changed, and the import now fails with an error message if the option is used in that situation. For other MySQL instances, the utility now checks whether the user has the required privileges to set the `sql_log_bin` system variable, and fails with an error message if they do not. (Bug #31748786)
- MySQL Shell's instance dump utility `util.dumpInstance()`, schema dump utility `util.dumpSchemas()`, and table dump utility `util.dumpTables()` ordered the data fetched for export using the first column of a unique index for the table. The same method was used to query data for chunking purposes. The utilities now use all columns of the unique index for ordering. In addition, performance is improved by the addition of a cache to store frequently-used instance metadata. The cache is populated for all the schema objects at once, rather than by individual queries as needed. (Bug #31706755)
- MySQL Shell's disconnect function was added to the `shell` global object. (Bug #31704380)

Changes in MySQL Shell 8.0.22 (2020-10-19, General Availability)

AdminAPI Bugs Fixed

- Due to the changes introduced in bug#31467823, AdminAPI user messages have been updated to use the same terminology. (Bug #31798724)

  References: See also: Bug #31462783, Bug #31467823.

- In 8.0.22, Group Replication deprecates the `group_replication_ip_whitelist` system variable in favor of `group_replication_ip_allowlist`. Therefore AdminAPI now includes a new option named `ipAllowlist` and deprecates the `ipWhitelist` option used with `dba.createCluster()`, *`Cluster`*`.addInstance()` and *`Cluster`*`.rejoinInstance()`.

Regardless of the target instance version, if `ipWhitelist` is used MySQL Shell prints a deprecation warning. When the target instance is 8.0.22 or later Group Replication reports a deprecation warning, but AdminAPI avoids this by ensuring the appropriate variable is set according to the version of MySQL running on the instance. If the target instance is 8.0.22 or later and the old `ipWhitelist` option is used, AdminAPI updates the new `group_replication_ip_allowlist` system variable. On instances running MySQL 8.0.21 or earlier, AdminAPI updates the older `group_replication_ip_whitelist` system variable.

This work also fixes a bug in *`Cluster`*`.options()`, which shows the list of all Group Replication options that are configurable by MySQL Shell. When an option is not set or does not exist (for example in a certain version) it is displayed as having a null value. However, when you passed in the `all` option to *`Cluster`*`.options()`, such variables were being excluded from the list. (Bug #31798495)

- The `dba.configureLocalInstance()` and `dba.configureInstance()` operations could not be used against instances that were part of unmanaged replication groups. This made it impossible to create a cluster administrator account, which is required when adopting a Group Replication group into an InnoDB Cluster. Creating cluster administrator accounts is very important for the remote management of InnoDB Clusters, and to avoid having to manually create the user and required privileges on each instance. The fix ensures that `dba.configureLocalInstance()` and `dba.configureInstance()` can be executed against instances belonging to unmanaged replication groups. Because such instances are ready for InnoDB Cluster usage, a message confirms this. (Bug #31691232)

- The `dba.rebootClusterFromCompleteOutage()` operation was only checking `GTID_EXECUTED` when validating for the instance that has the most transactions. Transactions that were received (and certified) but not executed yet were not included in that check. Also, transactions received through replication channels other than the Group Replication applier were also not being considered in that check, because the instance was probably the primary while the cluster was running. Ignoring these transactions led to data loss. Now, any known and managed replication channels are considered as part of the check. (Bug #31673163)

- The `dba.removeInstance()` operation failed if the instance being removed was not only unreachable but also unresolvable, which could be the case when the instance was running in a container that takes down its own DNS record when removed.

  Instances could not be removed even if force was enabled, because of a validation that ensured that the given instance is a valid address by checking if it is resolvable. If that validation failed, nothing else was attempted. The fix completely removes address resolution from the whole AdminAPI. That check was redundant, because invalid addresses would eventually lead to an error anyway when a connection was opened. IPv6 address syntax validations were left to remind you that `::1` should be specified as `[::1]`. (Bug #31632606)

- Calling *`Cluster`*`.rejoinInstance("`*`host`*`:`*`port`*`")` with no user name specified caused AdminAPI to try and connect using the operating system user name instead of the credentials used to connect the cluster object. Now, if credentials are not provided, they are taken from the target server's connection options. (Bug #31632554)

- When adding an instance to a cluster using MySQL Clone and monitoring the transfer of data, MySQL Shell could stop unexpectedly. This was due to the assumption that the Performance Schema would provide information on all four stages of cloning, which might not exist if the data set was very small. The fix ensures that updates are performed according to the information which is available at each time. (Bug #31545728)

- When an unreachable primary instance was forcibly removed from a cluster, the `group_replication_group_seeds` system variable was not updated because the group status was queried from the primary, which was missing. The cluster was left in an inconsistent state,

and if any of the instances were restarted, they would not be able to automatically rejoin, because `group_replication_group_seeds` contained an invalid address, which caused Group Replication to abort trying to join the group. (Bug #31531704)

- If the `validate_password` plugin was enabled, the `setupAdminAccount()` and `setupRouterAccount()` operations would fail with an error indicating the password did not meet the policy requirements. This happened regardless of whether the password met the policy requirements or not. (Bug #31491092)

- In the event of `dba.createCluster()` failing, the metadata record was left behind and Group Replication was also started. The cluster was left in an inconsistent state that could not be recovered from when calling `dba.createCluster()` again. Now, metadata changes during a `dba.createCluster()` operation are enclosed in a transaction, so that both the cluster and instance records are only committed if the operation succeeds. Group Replication is also stopped if the create operation does not complete successfully.

  Similarly, *`Cluster`*`.addInstance()` has been changed to ensure that its metadata record is only inserted when everything else has completed successfully. On retry, it skips any steps that would cause conflicts because Group Replication is already running. This introduces a behavior change, where adding an instance that is part of the group but not in the metadata just adds it to the metadata, instead of aborting and requiring you to execute *`Cluster`*`.rescan()`. (Bug #31455419)

- If a connection timed out during a *`Cluster`*`.status()` operation, MySQL Shell could appear to hang for a long time. To improve the responsiveness, the default timeout of AdminAPI operations has been reduced from 10 seconds to 2 seconds. This ensures operations like *`Cluster`*`.status()` do not appear to freeze for a long time when there are unreachable instances. (Bug #30884174)

- Instances operating in an InnoDB Cluster or InnoDB ReplicaSet are all required to have the same password for the administrative account. In a situation where the password on an instance joining a InnoDB Cluster or InnoDB ReplicaSet did not match the other instances, the error message did not explain this and the instance failed to join. Now, in such a situation the error is detected and the resulting message mentions that the password is the cause of the failure. It is recommended that you set up administrator accounts using the `setupAdminAccount()` operation, see Creating User Accounts for Administration. (Bug #30728744)

Functionality Added or Changed

- Two new utilities are available in MySQL Shell to export single tables from a MySQL server instance.

  - MySQL Shell's new table dump utility `util.dumpTables()` supports the export of a selection of tables or views from a schema, from an on-premise MySQL instance into an Oracle Cloud Infrastructure Object Storage bucket or a set of local files. It works in the same way as the instance dump utility `util.dumpInstance()` and schema dump utility `util.dumpSchemas()` introduced in 8.0.21, but with a different selection of suitable options. The exported items can then be imported into a MySQL Database Service DB System (a MySQL DB System, for short) or a MySQL Server instance using MySQL Shell's dump loading utility `util.loadDump()`.

  - MySQL Shell's new table export utility `util.exportTable()` exports a MySQL relational table into a data file in a variety of formats, either on the local server or in an Oracle Cloud Infrastructure Object Storage bucket. The data can then be uploaded into a table on a target MySQL server using MySQL Shell's parallel table import utility `util.importTable()`, which uses parallel connections to provide rapid data import for large data files. The data file can also be used to import data to a different application, or as a lightweight logical backup for a single data table.

- From MySQL Shell 8.0.22, when you export schemas to an Oracle Cloud Infrastructure Object Storage bucket using MySQL Shell's instance dump utility and schema dump utility, `util.dumpInstance()`

and `util.dumpSchemas()`, during the dump you can generate a pre-authenticated request URL for every item. The utilities do this by default when the `ocimds` option is set to true, and you can control the feature using the `ociParManifest` and `ociParExpireTime` options. The user account that runs MySQL Shell's dump loading utility `util.loadDump()` then uses the pre-authenticated request URLs to load the dump files without additional access permissions.

Bugs Fixed

- MySQL Shell's dump loading utility `util.loadDump()` would stop with an error if a data file's size was larger than an applicable server limit relating to the maximum transaction size, such as the `max_binlog_cache_size` limit. Now, the utility stops the data load in mid-file if the number of bytes uploaded is about to exceed 1.5 times the `bytesPerChunk` setting of the utility that created the data files, which is stored in the dump metadata. The data load is then restarted to upload the remainder of the file. Due to this new safeguard, the default `bytesPerChunk` setting of MySQL Shell's instance dump utility `util.dumpInstance()`, schema dump utility `util.dumpSchemas()`, and table dump utility `util.dumpTables()` has been increased from 32 MB to 64 MB. (Bug #31945539)

- MySQL Shell's instance dump utility `util.dumpInstance()` and schema dump utility `util.dumpSchemas()` did not take the row size into account when deciding whether to chunk table data, only the number of rows. A table with a smaller number of rows containing large amounts of data might therefore bypass chunking even if the row size exceeded the specified chunk size for the dump. The utilities now carry out chunking regardless of the number of rows, unless the data is estimated to fit in a single chunk of the specified size. (Bug #31938831)

- MySQL Shell's instance dump utility `util.dumpInstance()`, schema dump utility `util.dumpSchemas()`, and table dump utility `util.dumpTables()` use a chunking algorithm to split the data for a table into multiple files. It was possible for the algorithm to create a large number of very small files, causing issues with the dump. A minimum valid value is now specified, and the algorithm has been enhanced to ensure that the range being checked is always large enough to include sufficient rows. (Bug #31909408)

- When MySQL Shell's instance dump utility `util.dumpInstance()` or schema dump utility `util.dumpSchemas()` was splitting a table into chunks, an integer overflow, loop, and consequent out of memory error could occur if the maximum value in the index column was close to the maximum value of an integer. Extra checks have now been added to avoid this situation. (Bug #31896448)

- Corrupted SQL files could occur in MySQL Shell's dumps if the dumped string was exactly 2048 bytes, due to an internal module error which is now accounted for. (Bug #31843832)

- The administrative user account on an Oracle Cloud Infrastructure Compute instance, which was used when importing data with MySQL Shell's dump loading utility `util.loadDump()`, was unable to revoke privileges on MySQL system schemas (`mysql` and `sys`) that it did not have itself. Affected `REVOKE` statements are now stripped from an instance or schema dump created by MySQL Shell's instance dump utility `util.dumpInstance()` and schema dump utility `util.dumpSchemas()` when the `strip_restricted_grants` compatibility option is used. (Bug #31842532)

- MySQL Shell's instance dump utility and schema dump utility, dumpInstance() and dumpSchemas(), previously fetched the value of the `gtid_executed` system variable before the read lock was established, which could potentially lead to an inconsistency with the dumped data. The GTID set is now retrieved while the read lock is active. (Bug #31706940)

- From MySQL Shell 8.0.22, you can use the `–pym` command-line option to execute a specified Python module as a script in MySQL Shell's Python mode. `--pym` works in the same way as Python's `-m` command line option. (Bug #31694202)

- In MySQL Shell in Python mode, functions registered in JavaScript could not be called from global extension objects if they had optional arguments that were not provided. (Bug #31693096)
- MySQL Shell treated the character sequence */ as the end of a comment even when it was part of a quoted string. (Bug #31689135)
- MySQL Shell's instance dump utility and schema dump utility, dumpInstance() and dumpSchemas(), automatically select an index column to order and chunk the data. When an index with a functional key part was present for a table, the query for index columns returned `NULL` for the column name, causing an exception in the utility. These column names are now filtered out in the query. (Bug #31687059)
- MySQL Shell's parallel table import utility `util.importTable()` has a new option `decodeColumns`, and an enhancement to the columns option, to enable you to capture columns from the import file for input preprocessing (or to discard them) in the same way as with a `LOAD DATA` statement. The `decodeColumns` option specifies preprocessing transformations for the captured data in the same way as the `SET` clause of a `LOAD DATA` statement, and assigns them to columns in the target table. (Bug #31683641)
- MySQL Shell's dump loading utility loadDump() now verifies that it can open and write to the progress state file before it starts to retrieve the dump files, so that the utility does not spend time fetching the dump files if the import is subsequently going to fail for that reason. (Bug #31667539)
- Before MySQL 8.0, user accounts that have all privileges used the statement `GRANT ALL PRIVILEGES`, rather than the full list of privileges. When MySQL Shell's instance dump utility dumpInstance() was used to dump an instance, `ALL PRIVILEGES` was stripped from the statement, leaving the accounts with no privileges. The utility now replaces this grant with the administrator role. (Bug #31661180)
- MySQL Shell's instance dump utility and schema dump utility, dumpInstance() and dumpSchemas(), and dump loading utility loadDump(), previously timed out after the main thread had been idle for 8 hours. The timeout has now been extended indefinitely (to 1 year) so that long data load times do not cause the dump or import to fail. (Bug #31652265)
- MySQL Shell's dump loading utility loadDump() now includes a comment in its `LOAD DATA` statements to identify the data chunk that is currently being loaded. If you need to cancel the import, you can use this information to decide whether to let the import of this chunk complete or stop it immediately. (Bug #31646650)
- Previously, MySQL Shell's dump loading utility loadDump() closed DDL files after all tables from the schema had finished loading. For a dump with a very large number of DDL files, this could lead to unexplained runtime errors being returned. The utility now closes DDL files immediately after reading them. (Bug #31645896)
- MySQL Shell's dump loading utility loadDump() previously used only its main thread to execute the DDL scripts for tables. For a dump containing a large number of tables, fetching the DDL scripts could have a significant impact on the time taken. The utility now fetches and executes DDL scripts for tables using all its threads, with the exception of DDL scripts for views, which are fetched in parallel but executed only in the main thread to avoid race conditions. (Bug #31645806)
- MySQL Shell's dump loading utility loadDump() now loads views in sequence and only after all tables and placeholders from all schemas have been loaded, ensuring that views do not reference items that do not yet exist. (Bug #31645792)
- MySQL Shell now ignores any other available Python installations when a bundled compatible version of the Python interpreter has been installed. (Bug #31642521)

- When MySQL Shell's dump loading utility loadDump() is importing users and their roles and grants, an error is now returned if the user already exists in the target instance, and the user's grants from the dump files are not applied. Previously, the grants were applied to the existing user. (Bug #31627432)
- MySQL Shell's dump loading utility loadDump() now has an option `updateGtidSet` to apply the `gtid_executed` GTID set from the source MySQL instance to the `gtid_purged` GTID set on the target MySQL instance. You can append or replace the GTID set depending on the release of the target MySQL instance. (Bug #31627419)
- MySQL Shell's instance dump utility dumpInstance() and dump loading utility loadDump() now have options to include (`includeUsers`) or exclude (`excludeUsers`) named user accounts from the dump files or from the import. You can use these options to exclude user accounts that are not accepted for import to a MySQL DB System, or that already exist or are not wanted on the target MySQL instance. (Bug #31627292)
- With the `deferTableIndexes` option set to `all`, MySQL Shell's dump loading utility loadDump() defers creation of all secondary indexes until after the table is loaded. Previously, a table with a unique key column containing an auto-increment value failed to load in this situation. The utility now also creates indexes defined on columns with auto-increment values when the `deferTableIndexes` option is set to `all`. (Bug #31602690)
- The upload method used by MySQL Shell's instance dump utility `util.dumpInstance()` and schema dump utility `util.dumpSchemas()` to transfer files to an Oracle Cloud Infrastructure Object Storage bucket has a file size limit of 1.2 TiB. In MySQL Shell 8.0.21, the multipart size setting means that the numeric limit on multiple file parts applies first, creating a limit of approximately 640 GB. From MySQL Shell 8.0.22, the multipart size setting has been changed to allow the full file size limit. (Bug #31589858)
- MySQL Shell's upgrade checker utility checkForServerUpgrade() now checks for the obsolete `NO_AUTO_CREATE_USER` SQL mode. (Bug #31501981, Bug #99903)
- The parallelization of table loading by MySQL Shell's dump loading utility loadDump() has been improved. (Bug #31441903)
- If the settings for the server's global character set variables differed from the settings for the current session, MySQL Shell's parallel table import utility importTable() could not import data into a table created in the session whose name contained non-ASCII characters. Now, when the utility's characterSet option is specified, MySQL Shell executes a `SET NAMES` statement with the given value. (Bug #31412330)
- MySQL Shell's parallel table import utility importTable() could not import data if the global SQL mode `NO_BACKSLASH_ESCAPES` was set. The utility now clears the global SQL mode in sessions created to run the import. (Bug #31407133)
- When a function that was defined as a member of a MySQL Shell extension object had a number of optional parameters but no required parameters, in some situations calling the function with zero parameters or one parameter returned an error. (Bug #30744994)
- MySQL Shell's `db` global object did not return the current schema when its properties were queried. (Bug #30296825, Bug #96839)
- MySQL Shell has a new command `\disconnect`. The command disconnects MySQL Shell's global session (the session represented by the `session` global object) from the currently connected MySQL server instance, so that you can close the connection but still continue to use MySQL Shell. (Bug #28240416)

Changes in MySQL Shell 8.0.21 (2020-07-13, General Availability)

AdminAPI Added or Changed Functionality

- A new user configurable tag framework has been added to the metadata, to allow specific instances of a cluster or ReplicaSet to be marked with additional information. Tags can be any ASCII character and provide a namespace. You set tags for an instance using the `setInstanceOption()` operation. In addition, AdminAPI and MySQL Router 8.0.21 support specific tags, which enable you to mark instances as hidden and remove them from routing. MySQL Router then excludes such tagged instances from the routing destination candidates list. This enables you to safely take a server instance offline, so that applications and MySQL Router ignore it, for example while you perform maintenance tasks, such as server upgrade or configuration changes. To bring the instance back online, use the `setInstanceOption()` operation to remove the tags and MySQL Router adds the instance back to the routing destination candidates list, and it becomes online for applications. For more information, see Tagging the Metadata.

AdminAPI Bugs Fixed

- **Important Change:** Previously, Group Replication did not support binary log checksums, and therefore one of the requirements for instances in InnoDB Cluster was that binary log checksums were disabled by having the `binlog_checksum` system variable set to `NONE`. AdminAPI verified the value of `binlog_checksum` during the `dba.checkInstanceConfiguration()` operation and disallowed creating a cluster or adding an instance to a cluster that did not have binary log checksums disabled. In version 8.0.21, Group Replication has lifted this restriction, therefore InnoDB Cluster now permits instances to use binary log checksums, with `binlog_checksum` set to `CRC32`. The setting for `binlog_checksum` does not have to be the same for all instances. In addition, sandboxes deployed with version 8.0.21 and later do not set the `binlog_checksum` variable, which defaults to `CRC32`. (Bug #31329024)

- Adopting a Group Replication setup as a cluster can be performed when connected to any member of the group, regardless of whether it is a primary or a secondary. However, when a secondary member was used, `super_read_only` was being incorrectly disabled on that instance. Now, all operations performed during an adoption are done using the primary member of the group. This ensures that no GTID inconsistencies occur and that `super_read_only` is not incorrectly disabled on secondary members. (Bug #31238233)

- Using the `clusterAdmin` option to create a user which had a netmask as part of the host resulted in an error when this user was passed to the `dba.createCluster()` operation. Now, accounts that specify a netmask are treated as accounts with wildcards, meaning that further checks to verify if the account accepts remote connections from all instances are skipped. (Bug #31018091)

- The check for instance read-only compatibility was using a wrong MySQL version as the base version. The cross-version policies were added to Group Replication in version 8.0.17, but the check was considering instances running 8.0.16. This resulted in a misleading warning message indicating that the added instance was read-only compatible with the cluster, when this was not true (only for instances 8.0.16). The fix ensures that the check to verify if an instance is read-compatible or not with a cluster is only performed if the target instance is running version 8.0.17 or later. (Bug #30896344)

- The maximum number of instances in an InnoDB Cluster is 9, but AdminAPI was not preventing you from trying to add more instances to a cluster and the resulting error message was not clear. Now,

if a cluster has 9 instances, *Cluster*.addInstance prevents you adding more instances. (Bug #30885157)

- Adding an instance with a compatible GTID set to a InnoDB Cluster or InnoDB ReplicaSet on which provisioning is required should not require any interaction, because this is considered a safe operation. Previously, in such a scenario, when MySQL Clone was supported MySQL Shell still prompted to choose between cloning or aborting the operation. Now, the operation proceeds with cloning, because this is the only way to provision the instance.

  

  Note

  instances with an empty GTID set are not considered to have a compatible GTID set when compared with the InnoDB Cluster or InnoDB ReplicaSet. Such scenarios are considered to be unknown, therefore MySQL Shell prompts to confirm which action should be taken.

  (Bug #30884590)

- The Group Replication system variables (prefixed with group_replication) do not exist if the plugin has not been loaded. Even if the system variables are persisted to the instance's option file, they are not loaded unless the Group Replication plugin is also loaded when the server starts. If the Group Replication plugin is installed after the server starts, the option file is not reloaded, so all system variables have default values. Instances running MySQL 8.0 do not have a problem because SET PERSIST is used. However, on instances running version MySQL 5.7, the dba.rebootCluster() operation could not restore some system variables if the Group Replication plugin was uninstalled. Now, the dba.configureInstance() operation persists the Group Replication system variables to configuration files with the loose_ prefix. As a result, once the Group Replication plugin is installed, on instances running 5.7 the persisted values are used instead of the default values. (Bug #30768504)

- The updateTopologyMode option has been deprecated and the behavior of *Cluster*.rescan() has been changed to always update the topology mode in the Metadata when a change is detected. MySQL Shell now displays a message whenever such a change is detected. (Bug #29330769)

- The cluster.addInstance() and cluster.rejoinInstance() operations were not checking for the full range of settings which are required for an instance to be valid for adding to the cluster. This resulted in attempts to use instances which run on different operating systems to fail. For example, a cluster running on two instances that were hosted on a Linux based operating system would block the addition of an instance running Microsoft Windows. Now, cluster.addInstance() and cluster.rejoinInstance() operations validate the instance and prevent adding or rejoining an instance to the cluster if the value of the lower_case_table_names, group_replication_gtid_assignment_block_size or default_table_encryption of the instance are different from the ones on the cluster. (Bug #29255212)

Functionality Added or Changed

- MySQL Shell now has an instance dump utility, dumpInstance(), and schema dump utility, dumpSchemas(). The new utilities support the export of all schemas or a selected schema from an on-premise MySQL server instance into an Oracle Cloud Infrastructure Object Storage bucket or a set of local files. The schemas can then be imported into a MySQL Database Service DB System using MySQL Shell's new dump loading utility. The new utilities provide Oracle Cloud Infrastructure Object Storage streaming, MySQL Database Service compatibility checks and modifications, parallel dumping with multiple threads, and file compression. See Instance Dump Utility, Schema Dump Utility, and Table Dump Utility.

- MySQL Shell's new dump loading utility, loadDump(), supports the import of schemas dumped using MySQL Shell's new instance dump utility and schema dump utility into a MySQL Database Service

DB System. The dump loading utility provides data streaming from remote storage, parallel loading of tables or table chunks, progress state tracking, resume and reset capability, and the option of concurrent loading while the dump is taking place. See Dump Loading Utility.

- The X DevAPI implementation now supports JSON schema validation, which enables you to ensure that your documents have a certain structure before they can be inserted or updated in a collection. To enable or modify JSON schema validation you pass in a JSON object like:

```
{
    validation: {
      level: "off|strict",
      schema: "json-schema"
    }
}
```

Here, `validation` is JSON object which contains the keys you can use to configure JSON schema validation. The first key is `level`, which can take the value `strict` or `off`. The second key, `schema`, is a JSON schema, as defined at http://json-schema.org. If the `level` key is set to `strict`, documents are validated against the *`json-schema`* when they are added to the collection, or when an operation updates the document. If the document does not validate, the server generates an error and the operation fails. If the `level` key is set to `off`, documents are not validated against the *`json-schema`*.

You can pass a `validation` JSON object to the `schema.createCollection()` operation, to enable JSON schema validation, and `schema.modifyCollection()` operation, to change the current JSON schema validation, for example to disable validation. For more information, see JSON Schema Validation.

Bugs Fixed

- MySQL Shell plugins now support the use of the **kwargs syntax in functions defined in Python that are made available by the plugin. Using **kwargs in a function definition lets you call the function using a variable-length list of keyword arguments with arbitrary names. If the function is called from MySQL Shell's JavaScript mode, MySQL Shell passes the named arguments and their values into a dictionary object for the Python function. MySQL Shell first tries to associate a keyword argument passed to a function with any corresponding keyword parameter that the function defines, and if there is none, the keyword argument is automatically included in the **kwargs list. As a side effect of this support, any API function called from Python in MySQL Shell that has a dictionary of options as the last parameter supports defining these options using named arguments. (Bug #31495448)

- When switching to SQL mode, MySQL Shell queries the SQL mode of the connected server to establish whether the `ANSI_QUOTES` mode is enabled. Previously, MySQL Shell could not proceed if it did not receive a result set in response to the query. The absence of a result is now handled appropriately. (Bug #31418783, Bug #99728)

- In SQL mode, when the results of a query are to be printed in table format, MySQL Shell buffers the result set before printing, in order to identify the correct column widths for the table. With very large result sets, it was possible for this practice to cause an out of memory error. MySQL Shell now buffers a maximum of 1000 rows for a result set before proceeding to format and print the table. Note that if a field in a row after the first 1000 rows contains a longer value than previously seen in that column in the result set, the table formatting will be misaligned for that row. (Bug #31304711)

- Context switching in MySQL Shell's SQL mode has been refactored and simplified to remove SQL syntax errors that could be returned when running script files using the source command. (Bug #31175790, Bug #31197312, Bug #99303)

- The user account that is used to run MySQL Shell's upgrade checker utility checkForServerUpgrade() previously required `ALL` privileges. The user account now requires only the `RELOAD`, `PROCESS`, and `SELECT` privileges. (Bug #31085098)
- In Python mode, MySQL Shell did not handle invalid UTF-8 sequences in strings returned by queries. (Bug #31083617)
- MySQL Shell's parallel table import utility importTable() has a new option `characterSet`, which specifies a character set encoding with which the input data file is interpreted during the import. Setting the option to `binary` means that no conversion is done during the import. When you omit this option, the import uses the character set specified by the `character_set_database` system variable to interpret the input data file. (Bug #31057707)
- On Windows, if the MySQL Shell package was extracted to and used from a directory whose name contained multi-byte characters, MySQL Shell was unable to start. MySQL Shell now handles directory names with multi-byte characters correctly, including when setting up Python, loading prompt themes, and accessing credential helpers. (Bug #31056783)
- MySQL Shell's JSON import utility importJSON() now handles UTF-8 encoded files that include a BOM (byte mark order) at the start, which is the sequence `0xEF 0xBB 0xBF`. As a workaround in earlier releases, remove this byte sequence, which is not needed. (Bug #30993547, Bug #98836)
- When the output format was set to JSON, MySQL Shell's upgrade checker utility checkForServerUpgrade() included a description and documentation link for a check even if no issues were found. These are now omitted from the output, as they are with the text output format. (Bug #30950035)

Changes in MySQL Shell 8.0.20 (2020-04-20, General Availability)

- AdminAPI Added or Changed Functionality
- AdminAPI Bugs Fixed
- Functionality Added or Changed
- Bugs Fixed

AdminAPI Added or Changed Functionality

- MySQL Shell now enables you to create and configure the MySQL user accounts required by InnoDB Cluster, InnoDB ReplicaSet, and MySQL Router using AdminAPI operations. Previously, accounts required by InnoDB Cluster and InnoDB ReplicaSet had to be configured using the `clusterAdmin` option, and accounts required by MySQL Router had to be configured manually using SQL. The following AdminAPI operations are now available:
  - use *`Cluster`*`.setupAdminAccount(user, [options])` and *`Replicaset`*`.setupAdminAccount(user, [options])` to configure a MySQL user account with the necessary privileges to administer an InnoDB Cluster or InnoDB ReplicaSet.
  - use *`Cluster`*`.setupRouterAccount(user, [options])` and *`Replicaset`*`.setupRouterAccount(user, [options])` to create a MySQL user account or upgrade an existing account so that it that can be used by MySQL Router to operate on an InnoDB Cluster or InnoDB ReplicaSet. This is now the recommended method of adding MySQL Router accounts to use with InnoDB Cluster and InnoDB ReplicaSet.
- AdminAPI now uses a locking mechanism to avoid different operations from performing changes on an InnoDB ReplicaSet simultaneously. Previously, different instances of MySQL Shell could connect to

an InnoDB ReplicaSet at the same time and execute AdminAPI operations simultaneously. This could lead to inconsistent instance states and errors, for example if *ReplicaSet*.addInstance() and *ReplicaSet*.setPrimary() were executed in parallel.

Now, the InnoDB ReplicaSet operations have the following locking:

- dba.upgradeMetadata() and dba.createReplicaSet() are globally exclusive operations. This means that if MySQL Shell executes these operations on an InnoDB ReplicaSet, no other operations can be executed against the InnoDB ReplicaSet or any of its instances.
- *ReplicaSet*.forcePrimaryInstance() and *ReplicaSet*.setPrimaryInstance() are operations that change the primary. This means that if MySQL Shell executes these operations against an InnoDB ReplicaSet, no other operations which change the primary, or instance change operations can be executed until the first operation completes.
- *ReplicaSet*.addInstance(), *ReplicaSet*.rejoinInstance(), and *ReplicaSet*.removeInstance() are operations that change an instance. This means that if MySQL Shell executes these operations on an instance, the instance is locked for any further instance change operations. However, this lock is only at the instance level and multiple instances in an InnoDB ReplicaSet can each execute one of this type of operation simultaneously. In other words, at most one instance change operation can be executed at a time, per instance in the InnoDB ReplicaSet.
- dba.getReplicaSet() and *ReplicaSet*.status() are InnoDB ReplicaSet read operations and do not require any locking.

References: See also: Bug #30349849.

- Use the --replicaset option to configure MySQL Shell to work with an InnoDB ReplicaSet at start up. You must specify a connection to a replica set instance for this option to work correctly. If a replica set is found, this option populates the rs global object, which can then be used to work with the InnoDB ReplicaSet. As part of this addition, the --redirect-primary and --redirect-secondary options have been updated to also work with InnoDB ReplicaSet.

  When running MySQL Shell, use the shell.connectToPrimary([*connectionData*, *password*]) to check whether the target instance belongs to an InnoDB Cluster or InnoDB ReplicaSet. If so, MySQL Shell opens a new session to the primary, sets the global session to the established session and returns it. If no *connectionData* is provided, the current global session is used.

AdminAPI Bugs Fixed

- During distributed recovery which is using MySQL Clone, the instance restarts after the data files are cloned, but if the instance has to apply a large back log of transactions to finish the recovery process, then the restart process could take longer than the default 1 minute timeout. Now, when there is a large back log use the dba.restartWaitTimeout option to configure a longer timeout to ensure the apply process has time to process the transactions. (Bug #30866632)
- The dba.deleteSandboxInstance() operation did not provide an error if you attempted to delete a sandbox which did not exist. Now, in such a situation the dba.deleteSandboxInstance() operation throws a runtimeError. (Bug #30863587)
- The *Cluster*.forceQuorumUsingPartitionOf() operation was not stopping Group Replication on any reachable instances that were not part of the visible membership of the target instance, which could lead to undefined behavior if any of those instances were automatically rejoining the cluster. The fix stops Group Replication on any reachable instances that are not included in the new forced quorum membership. (Bug #30739252)

- It was possible for AdminAPI to select an invalidated instance as the latest primary, despite it having a lower `view_id`. This was because the process of getting the primary of the InnoDB ReplicaSet was incorrectly reconnecting to an invalidated member if it was the last instance in the InnoDB ReplicaSet. (Bug #30735124)

- When a cluster that was created with a MySQL Shell version lower than 8.0.19 was offline (for example after a server upgrade of the instances), if you then used MySQL Shell 8.0.19 to connect to the cluster, `dba.upgradeMetadata()` and `dba.rebootClusterFromCompleteOutage()` blocked each other. You could not run `dba.upgradeMetadata()` because it requires `dba.rebootClusterFromCompleteOutage()` to be run first to bring the cluster back online. And you could not run `dba.rebootClusterFromCompleteOutage()` because `dba.upgradeMetadata()` had not been run. To avoid this problem please upgrade to MySQL Shell 8.0.20, where the preconditions for `dba.rebootClusterFromCompleteOutage()` and `dba.forceQuorumUsingPartitionOf()` have been updated to ensure they are compatible with clusters created using earlier versions. In other words, they are available even if the metadata was created using an older MySQL Shell version. (Bug #30661129)

- Using MySQL Clone as the distributed recovery method to add an instance to an InnoDB ReplicaSet resulted in a segmentation fault if the target instance did not support `RESTART`. Now, the *`ReplicaSet`*`.addInstance()` operation aborts in such a situation and reverts changes after the connection timeout limit is reached. This is because the add operation needs to connect to the target instance to finish the operation. In such a situation, if it is not possible to upgrade the instance to a version of MySQL which supports `RESTART`, you have to restart the server manually, and then issue *`ReplicaSet`*`.addInstance()` again to retry. The retry can then use incremental recovery, which does not trigger the clone and the subsequent restart. (Bug #30657911)

- *`Cluster`*`.addInstance()` normally fails if there are errant GTIDs in the added instance, but if MySQL Clone is being used for distributed recovery, that check is bypassed because the cloning process fixes the problem. However, if all members are using IPv6, MySQL Clone cannot be used, so incremental recovery is used instead. In such a situation, the instance was being added to the cluster without errors, but the errant transaction persisted. Now, if all of the cluster's online instances are using IPv6 addresses and the operation tries to use MySQL Clone for distributed recovery, an error is thrown. (Bug #30645697)

- When an InnoDB Cluster or InnoDB ReplicaSet is using the MySQL Clone plugin, AdminAPI ensures the `performance_schema.clone_status` table is cleared out when the clone process starts. However, in some rare and very specific scenarios a race condition could happen and the clone operation was considered to be running before actually clearing out the table. In this situation, the MySQL Shell clone monitoring could result in an unexpected halt.

  As part of this fix, a potential infinite loop in the clone monitoring phase that could happen very rarely when the cloning process was extremely fast has also been fixed. (Bug #30645665)

- Group Replication system variable queries were being executed early, without considering whether the Group Replication plugin was installed yet. Now, the reboot operation has been fixed so that if system variable queries fail with `ER_UNKNOWN_SYSTEM_VARIABLE` then the Group Replication plugin is installed automatically. (Bug #30531848)

- The *`Cluster`*`.removeInstance(`*`instance`*`)` operation was not correctly handling the following cases:

  - if the instance had `report_host` set to a different value from the `instance_name` in the metadata, specifying the instance using its IP failed.

  - if the instance was unreachable, it was not possible to remove it if the given address did not match the address in the metadata.

  - when an instance was `OFFLINE` but reachable (for example because Group Replication stopped but the server was still running), *`Cluster`*`.removeInstance(`*`instance`*`)` failed. Now, in such a situation, if you are sure it is safe to remove the instance, use the `force=true` option, which means that synchronization is no longer attempted as part of the remove operation.
  - if the instance was `OFFLINE` but reachable, removing the instance through an address that did not match what was in the metadata would make the operation appear to succeed but the instance was not actually removed from the metadata.

  (Bug #30501628, Bug #30625424)

- After operations such as removing an instance or dissolving a cluster, the `group_replication_recovery` and `group_replication_applier` replication channels were not being removed. (Bug #29922719, Bug #30878446)
- The default location of the MySQL option file, for example `/etc/my.cnf`, stopped being detected by the `dba.configureInstance()` operation on some platforms (Debian and so on). This was a regression. The fix ensures that the predefined paths to option files matches the defaults, such as `/etc/my.cnf` and `/etc/mysql/my.cnf`. (Bug #96490, Bug #30171324)

Functionality Added or Changed

- A new method `shell.openSession` is provided in the `shell` global object to let you create and return a session object, rather than set it as the global session for MySQL Shell.
- You can now request compression for MySQL Shell connections that use X Protocol, as well as those that use classic MySQL protocol. For X Protocol connections, the default is that compression is requested, and uncompressed connections are allowed if the negotiations for a compressed connection do not succeed. For classic MySQL protocol connections, the default is that compression is disabled. After the connection has been made, the MySQL Shell `\status` command shows whether or not compression is in use for a session.

  New compression controls in MySQL Shell let you specify in the connection parameters whether compression is required, preferred, or disabled, select compression algorithms for the connection, and specify a numeric compression level for the algorithms.

Bugs Fixed

- When you create an extension object for MySQL Shell, the `options` key is no longer required when you specify a parameter of the data type "dictionary". If you do define options for a dictionary, MySQL Shell validates the options specified by the end user and raises an error if an option is passed to the function that is not in this list. If you create a dictionary with no list of options, any options that the end user specifies for the dictionary are passed directly through to the function by MySQL Shell with no validation. (Bug #30986260)
- A bug in MySQL Shell 8.0.19, affecting classic MySQL protocol connections only, meant that access was denied if a user had stored the connection's password with MySQL Shell and afterwards changed it. The password store now removes invalid passwords and presents the user with a password prompt as expected. (Bug #30912984, Bug #98503)
- When MySQL Shell's `\source` command was used in interactive mode to execute code from a script file, multi-line SQL statements in the script file could cause MySQL Shell to enter a loop of repeatedly executing the script. The issue has now been fixed. (Bug #30906751, Bug #98625)
- If a stored procedure was called in MySQL Shell but its result was not used, any subsequent SQL statement returned a result set error, and exiting MySQL Shell at that point resulted in an incorrect

shutdown. MySQL Shell cleared the first result set retrieved by a stored procedure in order to run a subsequent SQL statement, but did not check for any additional result sets that had been retrieved, which were left behind and caused the error. This check is now carried out and the additional result sets are discarded before another statement is executed. (Bug #30825330)

- Due to a regression in MySQL Shell 8.0.19, the upgrade checker utility `checkForServerUpgrade()` did not accept any runtime options if connection data was not provided as the first argument. The issue has been fixed and the utility's argument checking has been enhanced. (Bug #30689606)
- MySQL Shell, which now bundles Python 3.7.4, could not be built from source with Python 3.8. The incompatibilities have now been corrected so Python 3.8 may be used. (Bug #30640012)
- MySQL Shell's upgrade checker utility `checkForServerUpgrade()` did not flag removed system variables that were specified using hyphens rather than underscores. The utility also now continues with its sequence of checks if a permissions check cannot be performed at the required time. (Bug #30615030, Bug #97855)
- MySQL Shell's `\status` command showed that a connection was compressed if the connection had been created while starting MySQL Shell, but not if it was created after starting MySQL Shell. Compression is now shown in both cases. (Bug #29006903)

Changes in MySQL Shell 8.0.19 (2020-01-13, General Availability)

AdminAPI Added or Changed Functionality

- **Incompatible Change:** The AdminAPI now includes InnoDB ReplicaSets, that enables you to administer asynchronous replication set ups in a similar way to InnoDB Cluster. The addition of InnoDB ReplicaSet means that the InnoDB Cluster metadata schema has been upgraded to version 2.0. Regardless of whether you plan to use InnoDB ReplicaSet or not, to use MySQL Shell 8.0.19 and AdminAPI you must upgrade the metadata of your clusters. Connect MySQL Shell's global session to your cluster and use the new `dba.upgradeMetadata()` operation to upgrade the cluster's metadata to use the new metadata.

  

  Warning

  Without upgrading the metadata you cannot use MySQL Shell 8.0.19 to change the configuration of a cluster created with earlier versions. You can only read the configuration of the cluster, for example using *`Cluster`*`.status()`.

  The `dba.upgradeMetadata()` operation upgrades any automatically created MySQL Router users to have the correct privileges. Manually created MySQL Router users with a name not starting with `mysql_router_` are not automatically upgraded. This is an important step in upgrading your cluster, only then can the MySQL Router application be upgraded. See Upgrading an InnoDB Cluster.

  

  Warning

  A cluster which is using the new metadata cannot be administered by earlier MySQL Shell versions, for example once you upgrade to version 8.0.19 you can no longer use version 8.0.18 or earlier to administer the cluster.

To get information on which of the MySQL Router instances registered with a cluster require the metadata upgrade, issue `cluster.listRouters({'onlyUpgradeRequired':'true'})`.

- AdminAPI now supports socket connections to be used for cluster and replica set operations. (Bug #26265826)

- You can now get information about the MySQL Router instances registered with an InnoDB Cluster, and unregister a Router from a cluster, for example when you stop using it. Use the *`Cluster`*`.listRouters()` operation to show a list of all Routers registered with the cluster. The results provides information such as the hostname, ports, and so on.

  To filter the list to only show Router instances that do not support the latest metadata version use the `onlyUpgradeRequired` option, for example by issuing *`Cluster`*`.listRouters({'onlyUpgradeRequired':'true'})`. The returned information shows whether the Router instance is compatible or not with the Metadata version supported by the version of MySQL Shell you are using, which you can use when upgrading a cluster.

  To remove a registered Router from a cluster, use the *`Cluster`*`.removeRouterMetadata(`*`router`*`)` operation.

- The AdminAPI includes support for InnoDB ReplicaSet, that enables you to administer asynchronous replication sets in a similar way to InnoDB Cluster. InnoDB ReplicaSet enables you to deploy an asynchronous replication set consisting of a single primary and multiple secondaries (traditionally referred to as the MySQL replication master and slaves). You administer a ReplicaSet using AdminAPI operations, for example to check the status of the InnoDB ReplicaSet, and manually failover to a new primary in the event of a failure. Similar to InnoDB Cluster, MySQL Router supports bootstrapping against InnoDB ReplicaSet, which means you can automatically configure MySQL Router to use your InnoDB ReplicaSet without having to manually configure files. This makes InnoDB ReplicaSet a quick and easy way to get MySQL replication and MySQL Router up and running, making it well suited to scaling out reads, development environments, and applications that do not require the high availability offered by InnoDB Cluster. See Upgrading an InnoDB Cluster.

AdminAPI Bugs Fixed

- The `dba.configureLocalInstance()` operation could fail with a `key not found (LogicError)` error when executed on a non-sandbox instance where it did not have access to the `my.cnf` option file and the operation requested an output configuration file to be specified. (Bug #30657204)

- The extended status information now displays the version of metadata found on an InnoDB Cluster or InnoDB ReplicaSet. For example issue:

```
mysql-js> Cluster.status({extended=1})
mysql-js> ReplicaSet.status({extended=1})
```

  (Bug #30624615)

- If a cluster had been deployed with MySQL Shell version 8.0.14 or earlier, the metadata contained an invalid port number for X Protocol connections. The metadata upgrade catches such ports and removes the invalid number. To avoid problems with routing due to this incorrect port, upgrade your cluster's metadata. See Upgrading an InnoDB Cluster. (Bug #30618264)

- When a replication replica was configured to read from an InnoDB Cluster primary, even with the appropriate replication filtering to ignore the metadata replication was failing when an instance was added to the cluster using MySQL Clone as the recovery method. This was because the recovery process was granting a privilege on an account, which failed and broke replication. (Bug #30609075)

- The *Cluster*.removeInstance() operation was issuing a misleading error message when the instance was unreachable, indicating that it did not belong to the cluster when an alternative valid host or IP was used. Now, the error indicates that the instance is unreachable. (Bug #30580393)

- Although MySQL Shell version 8.0.18 added support for IPv6 in WL#12758, using an InnoDB Cluster which consisted of MySQL Shell version 8.0.18 and MySQL Router 8.0.18 with IPv6 addresses was not possible. With the release of version 8.0.19 of both MySQL Shell and MySQL Router, be aware that:

  - combining MySQL Shell version 8.0.18 with MySQL Router 8.0.18 causes Router to fail, due to no IPv6 support. (Bug#30354273)

  - combining MySQL Shell version 8.0.18 with Router 8.0.19 in a cluster which uses X Protocol connections, results in AdminAPI storing mysqlX IPv6 values incorrectly in the metadata, causing Router to fail. (Bug#30548843) However, combining MySQL Shell version 8.0.18 with Router 8.0.19 in a cluster which uses MySQL classic protocol connections, is possible.

  Therefore, to use InnoDB Cluster with IPv6 addresses, regardless of the protocol used, the recommended deployment is MySQL Shell 8.0.19 and MySQL Router 8.0.19. (Bug #30548843)

  References: See also: Bug #30354273.

- When using automatic rejoin, if a target instance was rejoining the cluster, operations such as dba.rebootClusterFromCompleteOutage(), *Cluster*.status(), and so on were failing. Now, clusters consider automatic rejoin as an instance state instead of a check that always aborts the operation. This ensures that *Cluster*.status() is reported even for instances which are rejoining the cluster, and that dba.rebootClusterFromCompleteOutage() can detect instances which are rejoining the cluster and override the rejoin operation so that the cluster can be properly rebooted. (Bug #30501590)

- SSL client certificate options for the clusterAdmin user were not being copied when setting up connection options, which made them fail when connecting. (Bug #30494198)

- When the automatically calculated localAddress is not valid, for example when it exceeds the valid range, the error message has now been improved. See Configuring InnoDB Cluster Ports. (Bug #30405569)

- The AdminAPI ensures that all members of a cluster have the same consistency level as configured at cluster creation time. However, when high and non-default consistency levels were chosen for the cluster, adding instances to it resulted in an error 3796 which indicates that group_replication_consistency cannot be used on the target instance. This happened because the consistency values of BEFORE, AFTER and BEFORE_AND_AFTER cannot be used on instances that are RECOVERING and several transactions happen while the instance is in the RECOVERING phase. Other AdminAPI commands result in the same error for the same scenario (high global consistency levels) whenever at least one member of the cluster is RECOVERING. For example, dba.getCluster(). The fix ensures that all sessions used by the AdminAPI use the consistency level of EVENTUAL when the cluster's consistency level is BEFORE, AFTER or BEFORE_AND_AFTER. (Bug #30394258, Bug #30401048)

- Some privileges required for persisting configuration changes on MySQL 8.0 servers were missing for the clusterAdmin users created by AdminAPI. In particular, an Access Denied error was being issued indicating the SYSTEM_VARIABLES_ADMIN and PERSIST_RO_VARIABLES_ADMIN privileges were required. Now these privileges are added for the clusterAdmin user on MySQL 8.0 servers. (Bug #30339460)

- When using MySQL Clone as the recovery method, trying to add an instance that did not support RESTART to a cluster caused MySQL Shell to stop unexpectedly. Now, in such a situation a message

explains that *`Cluster`*`.rescan()` must be used to ensure the instance is added to the metadata. (Bug #30281908)

- The `autocommit` and `sql_mode` system variables are session settings, but they can be set globally to different values. AdminAPI was failing if these variables had non-default values in several different ways, for example DML was failing, system variables could not be set and so on. (Bug #30202883, Bug #30324461)
- Attempting to bootstrap MySQL Router against an InnoDB Cluster which had had the cluster administration user modified or removed was failing. This was caused by the privileges granted on the InnoDB Cluster metadata table. The recommended solution is to upgrade to metadata 2.0, which changes the privileges on the metadata to ensure this issue does not occur. See Upgrading an InnoDB Cluster. (Bug #29868432)
- When you created a multi-primary cluster, the `group_replication_enforce_update_everywhere_checks` system variable was not being set automatically. However, switching to multi-primary mode automatically enables `group_replication_enforce_update_everywhere_checks` and switching to single-primary disables it. Now, the `dba.createCluster()` operation sets the `group_replication_enforce_update_everywhere_checks` variable as appropriate for single-primary or multi-primary clusters. (Bug #29794779)
- In version 8.0.16, the `autoRejoinTries` option was added to define the number of times an instance attempts to rejoin the cluster after being expelled. The option is a valid cluster setting, configurable through the AdminAPI like many other options. However, the `autoRejoinTries` option was not being listed by *`Cluster`*`.options()`. (Bug #29654346)
- The InnoDB Cluster metadata now supports host names up to 265 characters long, where 255 characters can be the host part and the remaining characters can be the port number. (Bug #29507913)
- `dba.createCluster()` could fail if the instance had been started with `innodb_default_row_format=COMPACT` or `innodb_default_row_format=REDUNDANT`. This was because no `ROW_FORMAT` was specified on the InnoDB Cluster metadata tables, which caused them to use the one defined in `innodb_default_row_format`. The metadata schema has been updated to use `ROW_FORMAT = DYNAMIC`. (Bug #28531271)
- When an instance restarted, for example after a complete outage, it could have `super_read_only` disabled. This meant that instances which were not the primary could be written to, resulting in the instances no longer being in synchrony. This could result in `dba.rebootClusterFromCompleteOutage()` failing with a `Conflicting transaction sets` error. The fix ensures that all instances have `super_read_only=1` persisted while they belong to the cluster, either through `SET PERSIST_ONLY`, or through `dba.configureLocalInstance()` for instances which do not support persisting. (Bug #97279, Bug #30545872)
- The *`Cluster`*`.status()` operation could report an error `get_uint(24): field value out of the allowed range` because it was always expecting a positive value for some fields that could in fact have negative values. For example, this could happen when the clocks of different instances were offset. (Bug #95191, Bug #29705983)
- If you changed the name of the `clusterAdmin` user once a cluster had been created, you could encounter an error such as `The user specified as a definer does not exist`. This was because the `clusterAdmin` user was used as the `DEFINER` of the views required by InnoDB Cluster, and if this user is renamed then the definer is in effect missing. In version 8.0.19 the InnoDB Cluster metadata has been changed to avoid this problem, use `dba.upgradeMetadata()` to upgrade the cluster. See Upgrading an InnoDB Cluster. Clusters deployed with 8.0.19 and later do not suffer from this issue. (Bug #92128, Bug #28541069)

- It was not possible to create a multi-primary cluster due to to cascading constraints on the InnoDB Cluster metadata tables. This has been fixed in version 8.0.19 and so to solve this issue upgrade your cluster using `dba.upgradeMetadata()`. See Upgrading an InnoDB Cluster. (Bug #91972, Bug #29137199)

Functionality Added or Changed

- The JavaScript function `require()` has been improved in MySQL Shell to support loading of local modules, in addition to built-in modules and modules that are on module search paths already known to MySQL Shell. If you specify the module name or path prefixed with `./` or `../`, MySQL Shell now searches for the specified module in the folder that contains the JavaScript file or module currently being executed, or in interactive mode, searches in the current working directory. If the module is not found in that folder, MySQL Shell proceeds to check the well-known module search paths specified by the sys.path variable.

- MySQL Shell's upgrade checker utility (the `util.checkForServerUpgrade()` operation) includes the following new and extended checks:

  - The utility now flags all `date`, `datetime`, and `timestamp` columns that have a default value of zero, and states if the SQL mode (either global or for the current session) allows the insertion of zero values for these column types. By default, these are no longer permitted in MySQL, and it is strongly advised to replace zero values with valid ones, as they might not work correctly in the future.

  - The check for usage of removed functions now includes the `PASSWORD()` function.

  - The utility now checks for any orphaned tables which are recognized by `InnoDB`, but the SQL layer thinks they are handled by a different storage engine. This situation can happen if manual updates are made in the data directory. Orphaned tables can stall the upgrade process if they are present.

- In MySQL Shell's interactive mode, for JavaScript, Python, or SQL, the `\source` command or its alias `\.` can be used to execute code from a script file at a given path. For compatibility with the `mysql` client, in SQL mode only, you can now execute code from a script file using the `source` command with no backslash and an optional SQL delimiter. `source` can be used both in MySQL Shell's interactive mode for SQL, to execute a script directly, and in a file of SQL code processed in batch mode, to execute a further script from within the file. In SQL mode only, you can also now use the `\source` command's alias `\.` (which does not use a SQL delimiter) in a file of SQL code processed in batch mode. So with MySQL Shell in SQL mode, you could now execute the script in the `/tmp/mydata.sql` file from either interactive mode or batch mode using any of these three commands:

  ```
  source /tmp/mydata.sql;
  source /tmp/mydata.sql
  \. /tmp/mydata.sql
  ```

  The command `\source /tmp/mydata.sql` is also valid, but in interactive mode only.

Bugs Fixed

- When searching for startup scripts in the platform's standard global configuration path (in the folder `%PROGRAMDATA%\MySQL\mysqlsh` on Windows, or `/etc/mysql/mysqlsh/` on Unix), MySQL Shell checked for the incorrect script name `shellrc`, rather than the correct name `mysqlshrc`. (Bug #30656548)

- On Windows, MySQL Shell passed UTF-8 encoded strings to the NTFS file system, which stores file names in UTF-16 encoding. The mismatch caused files and folders to be incorrectly named or located when non-ASCII characters were used. MySQL Shell now converts all strings used in operations on

NTFS from UTF-8 to UTF-16, and converts back to UTF-8 all strings received from Unicode function versions of Windows API calls. (Bug #30538516)

- Some host names were not parsed correctly in the connection data provided when running MySQL Shell's upgrade checker utility `checkForServerUpgrade()`, including where the user account's host name had been set up by specifying an IP address and subnet mask. (Bug #30536355, Bug #30696901, Bug #98056)
- If the MySQL Server environment variable `MYSQL_UNIX_PORT` (which specifies the default Unix socket file) was updated by the same process that was then used to create a MySQL Shell connection to a MySQL server using a socket file, MySQL Shell cached and connected using the socket file path that had previously been set, but reported that a connection had been made using the updated socket file path. The correct socket file used for the connection is now displayed. (Bug #30533318)
- If a prompt theme file used to customize the MySQL Shell prompt contained an ill-formed UTF-8 sequence, on startup an error message was displayed in place of the prompt text. MySQL Shell now validates the prompt theme file before loading it, and if there is a problem, uses a default prompt instead and issues an error message. (Bug #30406283)
- If MySQL Shell was installed on Microsoft Windows at a non-default location, and subsequently uninstalled, files created after installation by the Python library used by MySQL Shell were not removed. These files are now removed when MySQL Shell is uninstalled from any location. (Bug #30333801)
- Previously, most MySQL Shell options that expected an integer value could be set with an empty value, in which case the value 1 was applied. The exception was the `logLevel` option, which required a value. The behavior has now been standardized so all MySQL Shell options that expect a non-string value must be specified with a value, with the exception of options set on the command line. The affected options are `dba.gtidWaitTimeout`, `dba.logSql`, `history.maxSize`, and `verbose`. (Bug #30320839)
- When using MySQL Shell in interactive mode, using a template literal in a multiple-line JavaScript statement resulted in an error. The issue has now been fixed. (Bug #30248651)
- In Python mode, when multiple statements were input to MySQL Shell at the same time for execution in interactive mode, only the first statement was executed correctly. (Bug #30029568)
- The Debian control file for MySQL Shell has been corrected to remove packaging errors that occurred when certain variables were not defined. Thanks to Evgeniy Patlan for the fix. (Bug #29802600, Bug #95158)
- In MySQL Shell's parser for URI-like connection strings, handling of path separators was previously platform dependent. Unified parsing has now been introduced so that Windows named pipes can be parsed correctly on Unix platforms, and Unix socket files can be parsed correctly on Windows platforms. (Bug #29456981)
- MySQL Shell now looks up host names by obtaining the fully qualified domain name of the provided address and using the absolute form of this name (with a trailing dot). This method avoids potential issues caused by some network configurations that resolve host names as loopback addresses when they are actually addressable externally. (Bug #27704559)

Changes in MySQL Shell 8.0.18 (2019-10-14, General Availability)

- AdminAPI Added or Changed Functionality
- AdminAPI Bugs Fixed
- Functionality Added or Changed

- Bugs Fixed

AdminAPI Added or Changed Functionality

- MySQL Shell can now optionally log SQL statements that are executed by AdminAPI operations, and output them to the console if the `--verbose` option is set. The `dba.logSql` MySQL Shell configuration option or `--dba-log-sql` command line option activates logging for these statements. Statements executed by sandbox operations are excluded. Viewing the statements lets you observe the progress of the AdminAPI operations in terms of SQL execution, which can help with problem diagnosis for any errors.

- AdminAPI now supports IPv6 addresses if the target MySQL Server version is higher than 8.0.13. When using MySQL Shell 8.0.18 or higher, if all cluster instances are running 8.0.14 or higher then you can use an IPv6 or hostname that resolves to an IPv6 address for instance connection strings and with options such as `localAddress`, `groupSeeds` and `ipWhitelist`. For more information on using IPv6 see Support For IPv6 And For Mixed IPv6 And IPv4 Groups.

  References: See also: Bug #29557250, Bug #30111022, Bug #28982989.

- You can now reset the passwords for the internal recovery accounts created by InnoDB Cluster, for example to follow a custom password lifetime policy. Use the *`Cluster`*`.resetRecoveryAccountsPassword()` operation to reset the passwords for all internal recovery accounts used by the cluster. The operation sets a new random password for the internal recovery account on each instance which is ONLINE. If an instance cannot be reached, the operation fails. You can use the `force` option to ignore such instances, but this is not recommended, and it is safer to bring the instance back online before using this operation. This operation only applies to the passwords created by InnoDB cluster and cannot be used to update manually created passwords.

  

  Note

  The user which executes this operation must have all the required `clusterAdmin` privileges, in particular `CREATE USER`, in order to ensure that the password of recovery accounts can be changed regardless of the password verification-required policy. In other words, independent of whether the `password_require_current` system variable is enabled or not.

- MySQL Shell now supports specifying TLS version 1.3 and TLS cipher suites for classic MySQL protocol connections. You can use:

  - the `--tls-version` command option to specify TLS version 1.3.
  - the `--tls-ciphersuites` command option to specify cipher suites.
  - the `tls-versions` and `tls-ciphersuites` connection parameters as part of a URI-type connection string.

  

  Note

  `tls-versions` (plural) does not have a key-value equivalent, it is only supported in URI-type connection strings. Use `tls-version` to specify TLSv1.3 in a key-value connection string.

  To use TLS version 1.3, both MySQL Shell and MySQL server must have been compiled with OpenSSL 1.1.1 or higher. For more information see Using Encrypted Connections.

AdminAPI Bugs Fixed

- The `Cluster.rejoinInstance()` operation was not setting the auto increment values defined for InnoDB Cluster, leading to the use of the default Group Replication behavior if the instance configuration was not properly persisted, for example on 5.7 servers. The fix ensures that the `Cluster.rejoinInstance()` operation updates the auto increment settings of the target instance. (Bug #30174191)

- The output of `Cluster.status()` now includes the `replicationLag` field. The value is displayed in HH:MM:SS format and shows the time difference between the last transaction commit timestamp and the last transaction applied timestamp. This enables you to monitor the amount of time between the most recent transaction being committed and being applied on an instance. (Bug #30003034)

- `Cluster.addInstance()` did not ensure that the MySQL Clone plugin was installed or loaded on all cluster instances, when available and not disabled. This meant that whenever a cluster was created using an older MySQL Shell version, on a target MySQL instance supporting clone, the instance would not have the clone plugin installed. The result was that any `Cluster.addInstance()` call that used clone would fail. The same issue happened if an instance was added to a cluster consisting of one instance using the incremental recovery type and afterwards the seed instance was removed. This resulted in all cluster instances not having the clone plugin installed and therefore any instance added using the clone recovery method would fail. The fix ensures that the clone plugin is installed on all cluster members (if available and not disabled) at cluster creation time and also whenever an instance is added to a cluster. (Bug #29954085)

- The `Cluster.rejoinInstance()` operation was not checking the GTID consistency of an instance being rejoined to a cluster, which could result in data diverging. Now, the GTID consistency checks conducted as part of the `Cluster.rejoinInstance()` operation have been improved to check for irrecoverable or diverged data-sets and also for empty GTID sets. If an instance is found to not be consistent with the cluster, it is not rejoined and the operation fails with a descriptive error. You are also shown the list of errant transactions, possible outcomes and solutions. (Bug #29953812)

- `Cluster.describe()` was retrieving information about the cluster's topology and the MySQL version installed on instances directly from the current session. Now, the information is retrieved from the Metadata schema, and the MySQL version is not included in the information output by `Cluster.describe()`. (Bug #29648806)

- Using a password containing the ' character caused `dba.deploySandbox()` to fail. Now, all sensitive data is correctly wrapped to avoid such issues. (Bug #29637581)

- The `Cluster.addInstance()` operation creates internal recovery users which are required by the Group Replication recovery process. If the `Cluster.addInstance()` operation failed, for example because Group Replication could not start, the created recovery users were not removed. Now, in the event of a failure any internal users are removed. (Bug #25503159)

- When a cluster had lost quorum and the majority of the cluster instances were offline except the primary, after reestablishing quorum and adding a new instance to the cluster, it was not possible to remove and add the previous primary instance to the cluster. This was because the operation failed when trying to contact offline instances, which was because the feature to verify if a Group Replication protocol upgrade is required was not considering the possibility of some cluster instances being offline (not reachable). The fix improves the Group Replication protocol upgrade handling for the `Cluster.removeInstance()` operation, which now attempts to connect to other cluster instances and use the first reachable instance for this purpose. (Bug #25267603)

- The `dba.configureInstance()` operation was not setting the `binlog_checksum` option with the required value (NONE) in the option file for instances that did not support `SET PERSIST` (for example

instances running MySQL 5.7), when the option file path was not provided as an input parameter but instead specified though the operation wizard in interactive mode. (Bug #96489, Bug #30171090)

Functionality Added or Changed

- MySQL Shell's upgrade checker utility (the `util.checkForServerUpgrade()` operation) includes the following new and extended checks:

  - The utility now checks for tablespace names containing the string “FTS”, which can be incorrectly identified as tablespaces of full-text index tables, preventing upgrade. The issue has been fixed in MySQL 8.0.18, but affects upgrades to earlier MySQL 8.0 releases.

  - The check for database objects with names that conflict with reserved keywords now covers the additional keywords `ARRAY`, `MEMBER`, and `LATERAL`.

  - - The checks for obsolete `sql_mode` flags now check the global `sql_mode` setting.

  Running the upgrade checker utility no longer alters the `gtid_executed` value, meaning that the utility can be used on Group Replication group members without affecting their synchronization with the group. The upgrade checker also now works correctly with the `ANSI_QUOTES` SQL mode. (Bug #30002732, Bug #30103683, Bug #96351, Bug #30103640, Bug #96350)

  References: See also: Bug #29992589.

- MySQL Shell has two new built-in reports, which provide information drawn from various sources including MySQL's Performance Schema:

  - `threads` lists the current threads in the connected MySQL server which belong to the user account that is used to run the report. Using the report-specific options, you can choose to show foreground threads, background threads, or all threads. You can report a default set of information for each thread, or select specific information to include in the report from a larger number of available choices. You can filter, sort, and limit the output.

  - `thread` provides detailed information about a specific thread in the connected MySQL server. By default, the report shows information on the thread used by the current connection, or you can identify a thread by its ID or by the connection ID. You can select one or more categories of information, or view all of the available information about the thread.

  You can run the new reports using MySQL Shell's `\show` and `\watch` commands. The reports work with servers running all supported MySQL 5.7 and MySQL 8.0 versions. If any item of information is not available in the MySQL Server version of the target server, the reports leave it out.

- MySQL Shell has two new control commands:

  - The `\edit` (`\e`) command opens a command in the default system editor for editing. If you specify an argument to the command, this text is placed in the editor, and if you do not, the last command in the MySQL Shell history is placed in the editor. When you have finished editing, MySQL Shell presents your edited text ready for you to execute or cancel. The command can also be invoked using the short form `\e` or the key combination **Ctrl-X Ctrl-E**.

  - The `\system` (`\!`) command runs the operating system command that you specify as an argument to the command, then displays the output from the command in MySQL Shell. MySQL Shell returns an error if it was unable to execute the command.

- MySQL Shell now uses Python 3. For platforms that include a system supported installation of Python 3, MySQL Shell uses the most recent version available, with a minimum supported version of Python

3.4.3. For platforms where Python 3 is not included, MySQL Shell bundles Python 3.7.4. MySQL Shell maintains code compatibility with Python 2.6 and Python 2.7, so if you require one of these older versions, you can build MySQL Shell from source using the appropriate Python version.

Bugs Fixed

- In debug mode, MySQL Shell raised an assertion when handling a character contained in SQL strings. (Bug #30286680)
- If a Python lambda was added as a member of a MySQL Shell extension object, the Python object was not released correctly when MySQL Shell shut down, causing a segmentation fault. (Bug #30156304)
- A memory leak could occur when Python code was executed in interactive mode. (Bug #30138755)
- Help information for a MySQL Shell report could not be displayed unless there was an active session. MySQL Shell now checks for an open session only before actually running the report. (Bug #30083371)
- If a default schema was set for the MySQL Shell connection, and a different default schema was set after the connection was made, MySQL Shell's `\reconnect` command attempted to use the default schema from the original connection. The user's current default schema is now used for the reconnection attempt. (Bug #30059354)
- Due to a bug introduced by a change in MySQL Shell 8.0.16, the MSI file that is used by Windows Installer to install MySQL Shell overwrote the Windows PATH environment variable with the path to the application binary (mysqlsh), removing any other paths present. The issue has now been fixed. (Bug #29972020, Bug #95432)
- When the `\reconnect` command is used to attempt reconnection to a server, if the last active schema set by the user appears to be no longer available, MySQL Shell now attempts to connect with no schema set. (Bug #29954572)
- In interactive mode, MySQL Shell now handles multiline comments beginning with a slash and asterisk (/*) and ending with an asterisk and slash (*/). (Bug #29938424)
- The MySQL Shell `\source` command was not handled correctly when used in combination with SQL statements. (Bug #29889926)
- With MySQL Shell in SQL mode, if multiple SQL statements including a `USE` statement were issued on a single line with delimiters, the `USE` statement was not handled correctly. (Bug #29881615)
- If MySQL Shell's JSON import utility was used to send a large number of JSON documents to a server with insufficient processing capacity, the utility could fill up the write queue with batches of prepared documents, causing the connection to time out and the import to fail. The utility now waits to read the response from the server before sending the next batch of prepared documents to the server. (Bug #29878964)
- When MySQL Shell was built from source with a bundled OpenSSL package, the required linker flags were not set. The issue has now been fixed. (Bug #29862189)
- If a new query was executed in MySQL Shell while a result was still active, resulting in rows being cached, not all rows were returned by the old query. (Bug #29818714)

Changes in MySQL Shell 8.0.17 (2019-07-22, General Availability)

- AdminAPI Added or Changed Functionality
- AdminAPI Bugs Fixed
- Functionality Added or Changed

- Bugs Fixed

AdminAPI Added or Changed Functionality

- **Important Change:** The handling of internal recovery accounts created by InnoDB Cluster has been changed so that by default accounts are always created as “mysql_innodb_cluster_*server_id*@%”, where *server_id* is instance specific. This generated recovery account name is stored in the InnoDB Cluster metadata, to ensure the correct account is always removed if the instance is removed from the cluster.

  The previous behavior where multiple accounts would be created if `ipWhitelist` was given has been removed. In addition *`Cluster`*`.removeInstance()` no longer removes all recovery accounts on the instance being removed. It now removes the recovery account of the instance being removed on the primary and waits for the changes to be replicated before actually removing the instance from the group. Similarly, *`Cluster`*`.rejoinInstance()` no longer drops any recovery accounts. It only creates the recovery account of the instance being rejoined if it no longer exists on the primary (which it should in normal circumstances). If the recovery account already exists, it is reused by *`Cluster`*`.rejoinInstance()`.

  When a cluster is adopted from an existing Group Replication deployment, new recovery accounts are created and set for each member. Pre-existing accounts configured by the user are left unchanged and not dropped, unless they have the “mysql_innodb_cluster_” prefix.

  As part of this work, the behavior of `dba.createCluster()` and `Cluster.rebootClusterFromCompleteOutage()` operations has been changed. Now, if these operations encounter an instance which has `super_read_only=ON`, it is disabled automatically. Therefore the `clearReadOnly` option has been deprecated for these operations.

  References: See also: Bug #29629121, Bug #29559303.

- The `dba.createCluster()` operation has been improved, and as part of this work the order in which some steps of the operation are executed was changed. Now, the creation of the recovery (replication) user and updates to the Metadata are performed after bootstrapping the Group Replication group. As part of this work, the `dba.createCluster()` operation has been updated to support the `interactive` option, which is a boolean value that controls the wizards provided. When `interactive` is true, prompts and confirmations are displayed by the operation. The default value of `interactive` is equal to `useWizards` option.

- The compatibility policies that Group Replication implements for member versions in groups now consider the patch version of a member's MySQL Server release. Previously, when combining instances running different MySQL versions, only the major version was considered. InnoDB Cluster has been updated to support cluster operations where these compatibility policies have an impact. Using the patch version ensures better replication safety for mixed version groups during group reconfiguration and upgrade procedures. As part of this work the information provided about instances has been extended.

  The following InnoDB Cluster changes have been made to support the compatibility policies:

  - The *`Cluster`*`.addInstance()` operation now detects incompatibilities due to MySQL versions and in the event of an incompatibility aborts with an informative error.

  - The *`Cluster`*`.status()` attribute `mode` now considers the value of `super_read_only` and whether the cluster has quorum.

  - The *`Cluster`*`.status()` output now includes the boolean attribute `autoRejoinRunning`, which is displayed per instance belonging to the cluster and is true when automatic rejoin is running.

  - The `extended` option has been changed to accept integer or Boolean values. This makes the behavior similar to the `queryMembers` option, so that option has now been deprecated.

  References: See also: Bug #29557250.

- InnoDB Cluster supports the new MySQL Clone plugin on instances running 8.0.17 and later. When an InnoDB Cluster is configured to use MySQL Clone, instances which join the cluster choose whether to use Group Replication's distributed recovery or MySQL Clone to recover the transactions processed by the cluster. You can optionally configure this behavior, for example to force cloning, which replaces any transactions already processed. You can also configure how *`Cluster`*`.addInstance()` behaves, letting cloning operations proceed in the background or showing different levels of progress in MySQL Shell. This enables you to automatically provision instances in the most efficient way. In addition, the output of *`Cluster`*`.status()` for members in `RECOVERING` state has been extended to include recovery progress information to enable you to easily monitor recovery operations, whether they be using MySQL Clone or distributed recovery.

AdminAPI Bugs Fixed

- **Important Change:** The sandboxes deployed using the AdminAPI did not support the `RESTART` statement. Now, the wrapper scripts call `mysqld` in a loop so that there is a monitoring process which ensures that `RESTART` is supported. (Bug #29725222)

- The *`Cluster`*`.addInstance()` operation did not validate if the `server_id` of the joining instance was not unique among all cluster members. Although the use of a unique `server_id` is not mandatory for Group Replication to work properly (because all internal replication channels use `--replicate-same-server-id=ON`), it was recommended that all instances in a replication stream have a unique `server_id`. Now, this recommendation is a requirement for InnoDB Cluster, and when you use the *`Cluster`*`.addInstance()` operation if the `server_id` is already used by an instance in the cluster then the operation fails with an error. (Bug #29809560)

- InnoDB Clusters do not support instances that have binary log filters configured, but replication filters were being allowed. Now, instances with replication filters are also blocked from InnoDB Cluster usage. (Bug #29756457)

  References: See also: Bug #28064729, Bug #29361352.

- On instances running version 8.0.16, the *`Cluster`*`.rejoinInstance()` operation failed when one or more cluster members were in `RECOVERING` state, because the Group Replication communication protocol could not be obtained. More specifically, the `group_replication_get_communication_protocol()` User-Defined function (UDF) failed because it could only be executed if all members were `ONLINE`. Now, in the event of the UDF failing when rejoining an instance a warning is displayed and AdminAPI proceeds with the execution of the operation.

  Starting from MySQL 8.0.17, the `group_replication_get_communication_protocol()` UDF no longer issues an error if a member is `RECOVERING`. (Bug #29754915)

- On Debian-based hosts, `hostname` resolves to the IP address 127.0.1.1 by default, which does not match a real network interface. This is not supported by Group Replication, which made sandboxes deployed on such hosts unusable unless a manual change to the configuration file was made. Now, the sandbox configuration files created by MySQL Shell contain the following additional line:

  ```
  report_host = 127.0.0.1
  ```

  In other words the `report_host` variable is set to the loopback IP address. This ensures that sandbox instances can be used on Debian-based hosts without any additional manual changes. (Bug #29634828)

- If the binary logs had been purged from all cluster instances, `Cluster.checkInstanceState()` lacked the ability to check the instance's state, resulting in erroneous output values. Now, `Cluster.checkInstanceState()` validates the value of `GTID_PURGED` on all cluster instances and provides the correct output and also an informative message mentioning the possible actions to be taken. In addition, `Cluster.addInstance()` and `Cluster.rejoinInstance()` were not using the checks performed by `Cluster.checkInstanceState()` in order to verify the GTID status of the target instance in relation to the cluster. In the event of all cluster instances having their binary logs purged, the `Cluster.addInstance()` command would succeed but the instance would never be able to join the cluster as distributed recovery failed to execute. Now, both operations make use of the checks performed by `Cluster.checkInstanceState()` and provide informative error messages. (Bug #29630591, Bug #29790569)

- When using the `dba.configureLocalInstance()` operation in interactive mode, if you provided the path to an option file it was ignored. (Bug #29554251)

- Calling `cluster.removeInstance()` on an instance that did not exist, for example due to a typo or because it was already removed, resulted in a prompt asking whether the instance should be removed anyway, and the operation then failing. (Bug #29540529)

- To add or rejoin an instance to an existing InnoDB Cluster, the instance must not already be operating as a replica in asynchronous replication. Previously, `dba.checkInstanceConfiguration()` incorrectly reported target instances operating as a replica as valid for InnoDB Cluster usage. As a consequence, attempting to use the instance which had been incorrectly validated with operations such as `Cluster.addInstance()` failed without informative errors.

  Now, `dba.checkInstanceConfiguration()` verifies if the target instance is already configured as a replica and generates a warning if that is the case. Similarly, the `Cluster.addInstance()` and `Cluster.rejoinInstance()` operations detect such instances and block them from InnoDB Cluster usage, failing with an error. Note that this does not prevent instances which belong to a cluster also operating as the source in asynchronous replication. (Bug #29305551)

- The `dba.createCluster()` operation was allowed on a target instance that already had a populated Metadata schema, when the instance was already in that Metadata. The Metadata present on the target instance was being overridden, which was unexpected. Now, in such a situation the `dba.createCluster()` throws an exception and you can choose to either drop the Metadata schema or reboot the cluster. (Bug #29271400)

- When a sandbox instance of MySQL had been successfully started from MySQL Shell using `dba.startSandboxInstance()`, pressing **Ctrl+C** in the same console window terminated the sandbox instance. Sandbox instances are now launched in a new process group so that they are not affected by the interrupt. (Bug #29270460)

- During the creation of a cluster using the AdminAPI, some internal replication users are created with user names which start with “mysql_innodb_cluster”. However, if the MySQL server had a global password expiration policy defined, for example if `default_password_lifetime` was set to a value other than zero, then the passwords for the internal users expired after reaching the specified period. Now, the internal user accounts are created by the AdminAPI with password expiration disabled. (Bug #28855764)

- The `dba.checkInstanceConfiguration()` and `dba.configureInstance()` operations were not checking the validity of persisted configurations, which can be different from the corresponding system variable value, in particular when changed with `SET PERSIST_ONLY`. This could lead these operations to report wrong or inaccurate results, for example reporting that the instance configuration is correct when in reality the persisted configuration was invalid and wrong settings could be applied after a restart of the server, or inaccurately reporting that a server update was needed when only a restart was required. (Bug #28727505)

References: See also: Bug #29765093.

- When you removed an instance's metadata from a cluster without removing the metadata from the instance itself (for example because of wrong authentication or when the instance was unreachable) the instance could not be added again to the cluster. Now, another validation has been added to `*Cluster*.addInstance()` to verify if the instance already belongs to the cluster's underlying group but is not in the InnoDB Cluster metadata, issuing an error if it already belongs to the ReplicaSet. Similarly, an error is issued when the default port automatically set for the local address is invalid (out of range) instead of using a random port. (Bug #28056944)

- When issuing `dba.configureInstance()` in interactive mode and after selecting option number 2 “Create a new admin account for InnoDB cluster with minimal required grants” it was not possible to enter a password for the new user.

Functionality Added or Changed

- MySQL Shell has a new function for SQL query execution for X Protocol sessions that works in the same way as the function for SQL query execution in classic MySQL protocol sessions. The new function, `Session.runSql()`, can be used in MySQL Shell only as an alternative to X Protocol's `Session.sql()` to create a script that is independent of the protocol used for connecting to the MySQL server. Note that `Session.runSql()` is exclusive to MySQL Shell and is not part of the standard X DevAPI. As part of this change, the `ClassicSession.query` function for SQL query execution, which is a synonym of `ClassicSession.runSQL()`, is now deprecated.

  A new function `fetchOneObject()` is also provided for classic MySQL protocol and X Protocol sessions to return the next result as a scripting object. Column names are used as keys in the dictionary (and as object attributes if they are valid identifiers), and row values are used as attribute values in the dictionary. This function enables the query results to be browsed and used in protocol-independent scripts. Updates made to the returned object are not persisted on the database.

- MySQL Shell's new parallel table import utility provides rapid data import to a MySQL relational table for large data files. The utility analyzes an input data file, divides it into chunks, and uploads the chunks to the target MySQL server using parallel connections. The utility is capable of completing a large data import many times faster than a standard single-threaded upload using a `LOAD DATA` statement.

  When you invoke the parallel table import utility, you specify the mapping between the fields in the data file and the columns in the MySQL table. You can set field- and line-handling options as for the `LOAD DATA` command to handle data files in arbitrary formats. The default dialect for the utility maps to a file created using a `SELECT ... INTO OUTFILE` statement with the default settings for that statement. The utility also has preset dialects that map to the standard data formats for CSV files (created on DOS or UNIX systems), TSV files, and JSON, and you can customize these using the field- and line-handling options as necessary.

- MySQL Shell has a number of new display options for query results:

  - The `shell.dumpRows()` function can format a result set returned by a query in any of the output formats supported by MySQL Shell, and dump it to the console. Note that the result set is consumed by the function. This function can be used in MySQL Shell to display the results of queries run by scripts to the user in the same ways as the interactive SQL mode can.

  - The new MySQL Shell output format `json/array` produces raw JSON output wrapped in a JSON array. The output format `ndjson` is added as a synonym for `json/raw`, and both those output formats produce raw JSON output delimited by newlines. You can select MySQL Shell output formats by starting MySQL Shell with the `--result-format=[value]` command line option, or setting the MySQL Shell configuration option `resultFormat`.

A new function `shell.unparseUri()` is also added, which converts a dictionary of URI components and connection options into a valid URI string for connecting to MySQL.

- You can now extend MySQL Shell with plugins that are loaded at startup. MySQL Shell plugins can be written in either JavaScript or Python, and the functions they contain are available in MySQL Shell in both JavaScript and Python modes. The plugins can be used to contain functions that are registered as MySQL Shell reports, and functions that are members of extension objects that are made available as part of user-defined MySQL Shell global objects.

  You can create a MySQL Shell plugin by storing code in a subfolder of the `plugins` folder in the MySQL Shell user configuration path, with an initialization file that MySQL Shell locates and executes at startup. You can structure a plugin group, with a collection of related plugins that can share common code, by placing the subfolders for multiple plugins in a containing folder under the `plugins` folder.

- You can now extend the base functionality of MySQL Shell by defining extension objects and making them available as part of additional MySQL Shell global objects. Extension objects can be written in JavaScript or Python. When you create and register an extension object, it is available in MySQL Shell in both JavaScript and Python modes. You construct and register extension objects using functions provided by the built-in global object `shell`.

- You can now configure MySQL Shell to send logging information to the console, in addition to sending it to the application log. The `--verbose` command-line option and the `verbose` MySQL Shell configuration option activate this function. By default, when the option is set, internal error, error, warning, and informational messages are sent to the console, which is the equivalent to a logging level of 5 for the application log. You can add three further levels of debug messages, up to the highest level of detail.

- MySQL Shell's upgrade checker utility (the `util.checkForServerUpgrade()` operation) carries out two new checks. When checking for upgrade from any MySQL 5.7 release to any MySQL 8.0 release, the utility identifies partitioned tables that use storage engines other than InnoDB or NDB and therefore rely on generic partitioning support from the MySQL server, which is no longer provided. When checking for upgrade from any release to MySQL 8.0.17, the utility identifies circular directory references in tablespace data file paths, which are no longer permitted.

- X DevAPI now supports indexing array fields. A single index field description can contain a new member name `array` that takes a Boolean value. If set to true, the field is assumed to contain arrays of elements of the given type. In addition, the set of possible index field data types (used as values of member type in index field descriptions) is extended with type `CHAR(N)`, where the length N is mandatory.

- MySQL Shell now supports the ability to send connection attributes (key-value pairs that application programs can pass to the server at connect time). MySQL Shell defines a default set of attributes, which can be disabled or enabled. In addition, applications can specify attributes to be passed in addition to the default attributes. The default behavior is to send the default attribute set.

  You specify connection attributes as a `connection-attributes` parameter in a connection string. The `connection-attributes` parameter value must be empty (the same as specifying `true`), a `Boolean` value (`true` or `false` to enable or disable the default attribute set), or a list or zero or more `key=value` specifiers separated by commas (to be sent in addition to the default attribute set). Within a list, a missing key value evaluates as an empty string. Examples:

```
"mysqlx://user@host?connection-attributes"
"mysqlx://user@host?connection-attributes=true"
"mysqlx://user@host?connection-attributes=false"
"mysqlx://user@host?connection-attributes=[attr1=val1,attr2,attr3=]"
```

```
"mysqlx://user@host?connection-attributes=[]"
```

You can specify connection attributes for both X Protocol connections and classic MySQL protocol connections. The default attributes set by MySQL Shell are:

```
> \sql SELECT ATTR_NAME, ATTR_VALUE FROM performance_schema.session_account_connect_attrs;
+-----------------+------------+
| ATTR_NAME       | ATTR_VALUE |
+-----------------+------------+
| _pid            | 28451      |
| _platform       | x86_64     |
| _os             | Linux      |
| _client_name    | libmysql   |
| _client_version | 8.0.17     |
| program_name    | mysqlsh    |
+-----------------+------------+
```

Application-defined attribute names cannot begin with `_` because such names are reserved for internal attributes.

If connection attributes are not specified in a valid way, an error occurs and the connection attempt fails.

For general information about connection attributes, see Performance Schema Connection Attribute Tables.

- MySQL Shell now supports the `OVERLAPS` and `NOT OVERLAPS` operators for expressions on JSON arrays or objects:

```
expr OVERLAPS expr
expr NOT OVERLAPS expr
```

These operators behave in a similar way to the `JSON_OVERLAPS()` function. Suppose that a collection has these contents:

```
mysql-js> myCollection.add([{ "_id": "1", "list": [1, 4] }, { "_id": "2", "list": [4, 7] }])
```

This operation:

```
mysql-js> var res = myCollection.find("[1, 2, 3] OVERLAPS $.list").fields("_id").execute();
mysql-js> res
```

Should return:

```
{
"_id": "1"
}
1 document in set (0.0046 sec)
```

This operation:

```
mysql-js> var res = myCollection.find("$.list OVERLAPS [4]").fields("_id").execute();
mysql-js> res
```

Should return:

```
{
"_id": "1"
}
{
"_id": "2"
}
2 documents in set (0.0031 sec)
```

An error occurs if an application uses either operator and the server does not support it.

Bugs Fixed

- With MySQL Shell in Python mode, using auto-completion on a native MySQL Shell object caused informational messages about unknown attributes to be written to the application log file. (Bug #29907200)
- The execution time for statements issued in MySQL Shell in multiple-line mode has been reduced by reparsing the code only after the delimiter is found. (Bug #29864587)
- Python's `sys.argv` array was only initialized when MySQL Shell was started in batch mode, and was not initialized when MySQL Shell was started in interactive mode. (Bug #29811021)
- MySQL Shell incorrectly encoded the `CAST` operation as a function call rather than a binary operator, resulting in SQL syntax errors. (Bug #29807711)
- MySQL Shell now supports the unquoting extraction operator `->>` for JSON. (Bug #29794340)
- Handling of empty lines in scripts processed by MySQL Shell in batch mode has been improved. (Bug #29771369)
- On Windows, when a MySQL Shell report was displayed using the `\watch` command, pressing **Ctrl+C** to interrupt execution of the command did not take effect until the end of the refresh interval specified with the command. The interrupt now takes effect immediately. Also, any queries executed by reports run using the `\show` or `\watch` commands are now automatically cancelled when **Ctrl+C** is pressed. (Bug #29707077)
- In Python mode, native dictionary objects created by MySQL Shell did not validate whether they contained a requested key, which could result in random values being returned or in a `SystemError` exception being thrown. Key validation has now been added, and a `KeyError` exception is thrown if an invalid key is requested. (Bug #29702627)
- When using MySQL Shell in interactive mode, if raw JSON output was being displayed from a source other than a terminal (for example a file or a pipe), in some circumstances the prompt was shown on the same line as the last line of the output. The issue has now been corrected, and a new line is printed before the prompt message if the last line of the output did not end with one. (Bug #29699640)
- The MySQL Shell `\sql` command, which executes a single SQL statement while another language is active, now supports the `\G` statement delimiter to print result sets vertically. (Bug #29693853)
- Some inconsistencies in MySQL Shell's choice of `stdout` or `stderr` for output have been corrected, so that only expected output that is intended to be processed by other programs goes to `stdout`, and all informational messages, warnings, and errors go to `stderr`. (Bug #29688637)
- When MySQL Shell was started with the option `--quiet-start=2` to print only error messages, warning messages about the operation of the upgrade checker utility `checkForServerUpgrade()` were still printed. (Bug #29620947)
- In Python mode, native dictionary objects created by MySQL Shell did not provide an iterator, so it was not possible to iterate over them or use them with the `in` keyword. Functionality to provide Python's iterator has now been added. (Bug #29599261)
- When a MySQL Shell report was displayed using the `\watch` command, the screen was cleared before the report was rerun. With a report that executed a slow query, this resulted in a blank screen being displayed for noticeable periods of time. The screen is now cleared just before the report generates its first text output. (Bug #29593246)

- MySQL Shell's upgrade checker utility `checkForServerUpgrade()` returned incorrect error text for each removed system variable that was detected in the configuration file. (Bug #29508599)
- MySQL Shell would hang when attempting to handle output from a stored procedure that produced results repeatedly from a single statement. The issues have now been corrected. (Bug #29451154, Bug #94577, Bug #28880081, Bug #93070)
- You can now specify the command line option `--json` to activate JSON wrapping when you start MySQL Shell to use the upgrade checker utility. In this case, JSON output is returned as the default, and you can choose raw JSON format by specifying `--json=raw`. Also, warning and error messages relating to running the utility have been removed from the JSON output. (Bug #29416162)
- In SQL mode, when MySQL Shell was configured to use an external pager tool to display output, the pager was invoked whether or not the query result was valid. For an invalid query, this resulted in the pager displaying an empty page, and the error message was only visible after quitting the pager. The pager tool is now only invoked when a query returns a valid result, otherwise the error message is displayed. (Bug #29408598, Bug #94393)
- MySQL Shell did not take the `ANSI_QUOTES` SQL mode into account when parsing quote characters. (Bug #27959072)
- Prompt theme files for MySQL Shell that were created on Windows could not be used on other platforms. The issue, which was caused by the parser handling the carriage return character incorrectly, has now been fixed. (Bug #26597468)
- The use of the `mysqlsh` command-line option `--execute` (`-e`) followed by `--file` (`-f`) when starting MySQL Shell is now disallowed, as these options are mutually exclusive. If the options are specified in that order, an error is returned. Note that if `--file` is specified first, `--execute` is treated as an argument of the processed file, so no error is returned. (Bug #25686324)
- Syntax errors returned by MySQL Shell's JavaScript expression parser have been improved to provide context and clarify the position of the error. (Bug #24916806)

Changes in MySQL Shell 8.0.16 (2019-04-25, General Availability)

AdminAPI Added or Changed Functionality

- MySQL InnoDB cluster automatically and transparently manages the communication protocol versions of its members, whenever the cluster topology is changed using AdminAPI operations. An InnoDB cluster always uses the most recent communication protocol version that is supported by all instances that are part of the cluster or joining it.
  - When an instance is added to, removed from, or rejoins the cluster, or a rescan or reboot operation is carried out on the cluster, the communication protocol version is automatically set to a version supported by the instance that is now at the earliest MySQL Server version.
  - When you carry out a rolling upgrade by removing instances from the cluster, upgrading them, and adding them back into the cluster, the communication protocol version is automatically upgraded when the last remaining instance at the old MySQL Server version is removed from the cluster prior to its upgrade.

To see the communication protocol version in use in an InnoDB cluster, use the *Cluster*.status() function with the 'extended' option enabled. The communication protocol version is returned in the 'GRProtocolVersion' field, provided that the cluster has quorum and no cluster members are unreachable.

- The new autoRejoinTries option enables you to configure how many times an instance tries to rejoin a group after being expelled. In scenarios where network glitches happen but recover quickly, setting this option prevents you from having to manually add the expelled instance back to the group. The autoRejoinTries option accepts positive integer values between 0 and 2016 and the default value is 0, which means that instances do not try to automatically rejoin. Set the value to a valid integer to configure the number of attempts expelled instances should make to rejoin the group. You can pass the autoRejoinTries option to these AdminAPI operations:

  - dba.createCluster()
  - *Cluster*.addInstance()
  - *Cluster*.setOption()
  - *Cluster*.setInstanceOption()

  When you configure the autoRejoinTries option, it sets the group_replication_autorejoin_tries system variable. Passing the option to dba.createCluster(), *Cluster*.addInstance() or *Cluster*.setInstanceOption() configures the automatic rejoin for specific cluster instances. Passing the option to *Cluster*.setOption() configures the automatic rejoin for all cluster instances.

  For more information, see Responses to Failure Detection and Network Partitioning.

- AdminAPI now reports information about the version of MySQL running on instances. This information is available from the following operations:

  - *Cluster*.status()
  - *Cluster*.describe()
  - *Cluster*.rescan()

  See Checking the MySQL Version on Instances for more information.

AdminAPI Bugs Fixed

- Removing an instance from a cluster when the instance to be removed had no user defined for the group_replication_recovery channel resulted in dropping users on the remaining instances of the cluster. (Bug #29617572)
- The failoverConsistency option has been deprecated and a new option named consistency has been added, to make it more consistent with the target Group Replication group_replication_consistency system variable name. The MySQL Shell online documentation now also correctly describes all of the values you can assign to the consistency option. (Bug #29356599)
- The dba.configureLocalInstance() operation would remove any section that did not start with mysqld from the provided option file. This could remove sections such as the client section from the option file. (Bug #29349014)

- When an instance with X Plugin disabled was added to an InnoDB Cluster, if the instance was later removed from the cluster using *`Cluster`*`.removeInstance()` the operation failed with `LogicError "get_string(7): field is NULL"`. This was a regression introduced by the fix for Bug#27677227. (Bug #29304183)

- There was an inconsistency between the behavior of `dba.checkInstanceConfiguration()` and the commands to add instances to the cluster (`dba.createCluster()` and *`Cluster`*`.addInstance()`) regarding the localhost and loopback address validation. In particular, a simple error was printed by `dba.checkInstanceConfiguration()` but the execution of the operation continued showing that everything was correct at the end of the operation, while an error was issued and the execution stopped for `dba.createCluster()` and `Cluster.addInstance()`.

  As part of fixing this issue, it was decided that the existing localhost and loopback address validations are no longer needed and should be removed. In particular, whatever address is specified for `report_host`, even if it is localhost or the loopback address (127.0.0.1), should be allowed, because it was explicitly specified by the user to use it. (Bug #29279941)

- The `dba.rebootClusterFromCompleteOutage()` operation was not preserving the existing Group Replication configurations previously set for the instances. In particular, the Group Replication local address and exit state action values were being changed. Now all settings are read at the time of rebooting the cluster. (Bug #29265869)

- Using either *`Cluster`*`.setOption()` or *`Cluster`*`.setInstanceOption()` to set an option which only exists in MySQL 8.0 on an instance running MySQL 5.7 was not being caught correctly. (Bug #29246657)

- On Debian-based platforms (such as Ubuntu), if the hostname resolved to 127.0.1.1 - which is the default on these platforms - it was not possible to create a cluster using the default settings. Now, in such situations a proper validation of the instance is performed before creating a cluster and adding instances to it. (Bug #29246110)

- The `dba.checkInstanceConfiguration()` operation did not validate host restrictions for the account provided for cluster administration, for example if the account could actually connect to all of the instances in the cluster. In particular, now an error is issued if the provided user account is only able to connect through localhost. (Bug #29018457)

- InnoDB Cluster configured `auto_increment_increment` and `auto_increment_offset` on instances for clusters running in multi-primary mode and consisting of up to 7 instances based on the logic described at InnoDB Cluster and Auto-increment. But Group Replication permits groups to contain up to 9 members, and *`Cluster`*`.addInstance()` and *`Cluster`*`.removeInstance()` were not following the logic used for other operations. Now, InnoDB Cluster uses the same logic for auto increment regardless of the operation used and correctly handles multi-primary clusters with more than 7 instances. (Bug #28812763)

- MySQL Shell uses the host value of the provided connection parameters as the target hostname used for AdminAPI operations, namely to register the instance in the metadata (for the `dba.createCluster()` and `cluster.addInstance()` operations). However, the host used for the connection parameters might not match the hostname that is used or reported by Group Replication, which uses the value of the `report_host` system variable when it is defined (in other words it is not `NULL`), otherwise the value of `hostname` is used. Therefore, AdminAPI now follows the same logic to register the target instance in the metadata and as the default value for the `group_replication_local_address` variable on instances, instead of using the host value from the instance connection parameters. During this fix it was detected that when the `report_host` variable was set to empty, Group Replication uses an empty value for the host but AdminAPI (for example in commands such as `dba.checkInstanceConfiguration()`, `dba.configureInstance()`, `dba.createCluster()`) reports the hostname as the value used which is inconsistent with the value

reported by Group Replication. An error is now issued by AdminAPI if an empty value is set for the `report_host` system variable. (Bug #28285389)

- In the event that `dba.createCluster()` failed and a rollback was performed to remove the created replication (recovery) users, the account created at `localhost` and any of the `ipWhitelist` addresses were not being removed. The fix ensures that the replication accounts are removed whenever a rollback related to `dba.createCluster()` is performed. This work was based on a code contribution from Bin Hong. (Bug #94182, Bug #29308037)

Functionality Added or Changed

- **Important Change:** Attempting to connect to an X Protocol port, 33060 by default, using the classic MySQL protocol resulted in the following error: `ERROR 2013 (HY000): Lost connection to MySQL server at 'reading initial communication packet', system error: 0`

  This was because of differences in X Protocol and classic MySQL protocol clients expectations on how connections were initialized. Now, in such a situation the generated error message is `ERROR 2007 (HY000): Protocol mismatch; server version = 11, client version = 10`. If you encounter this error then you are probably trying to use the wrong port for the protocol your client is using.

  As part of this improvement the `mysqlx_enable_hello_notice` system variable has been added, which controls messages sent to classic MySQL protocol clients that try to connect over X Protocol. When enabled, clients which do not support X Protocol that attempt to connect to the server X Protocol port receive an error explaining they are using the wrong protocol. Set `mysqlx_enable_hello_notice` to false to permit clients which do not recognize the hello message to still connect.

- MySQL Shell's upgrade checker utility can now check the configuration file (`my.cnf` or `my.ini`) for the server instance. The utility checks for any system variables that are defined in the configuration file but have been removed in the target MySQL Server release, and also for any system variables that are not defined in the configuration file and will have a different default value in the target MySQL Server release. For these checks, when you invoke `checkForServerUpgrade()`, you must provide the file path to the configuration file. If you omit the file path and the upgrade checker utility needs to run a check that requires the configuration file, that check fails with a message informing you that you must specify the file path. (Bug #27801824, Bug #29222179)

- MySQL Shell now has a framework and commands that you can use to set up and run reports to display live information from a MySQL server, such as status and performance information. Reports can be run once using the MySQL Shell `\show` command, or run then refreshed continuously in a MySQL Shell session using the `\watch` command. They can also be accessed as API functions in the `shell.reports` object.

  The reporting facility supports both built-in reports and user-defined reports. User-defined reports can be created in the supported scripting languages JavaScript and Python, and can be run in any MySQL Shell mode (JavaScript, Python, or SQL), regardless of the language that the report was written in. Reports can be saved in a folder in the MySQL Shell configuration path and automatically loaded at startup. You can also create a report directly in the MySQL Shell prompt. You register a report to MySQL Shell using the `shell.registerReport` method to provide information about the report and the options and arguments that it supports.

  For more information, see Reporting with MySQL Shell.

- When running MySQL Shell in interactive mode, you can now execute an SQL statement without switching to SQL mode and back again afterwards. This function enables you to conveniently issue

some SQL statements in the context of a longer AdminAPI workflow in JavaScript or Python mode. Use the `\sql` command immediately followed by the SQL statement, for example:

```
\sql select * from sakila.actor limit 3;
```

The SQL statement does not need any additional quoting, and the statement delimiter is optional. With this format, MySQL Shell does not switch mode as it would if you entered the `\sql` command. After the SQL statement has been executed, MySQL Shell remains in JavaScript or Python mode.

You cannot use multiple line mode when you use the `\sql` command with a query to execute single SQL statements while another language is active. The command only accepts a single SQL query on a single line.

- MySQL Shell history is now split per active language which the command was issued under. This means that your history now matches the active language, for example when you are running in JavaScript mode having issued `\js`, the history contains the previous JavaScript statements you issued, and when you issue `\sql` to change to SQL mode your history contains the previous SQL statements you issued. Similarly, now any history related commands such as `\history clear` or `\history delete` are performed on the history of the current active language. When you install this version, any existing MySQL Shell history files are duplicated to ensure that existing history is not lost. Subsequent operations are then added to the language specific history file.

- When `resultFormat` was set to `json` or `json/raw`, every result was being returned as a JSON document. This behavior was expected when JSON wrapping is off (in other words the `--json` command option was not used when starting MySQL Shell). Now, for consistency reasons when JSON wrapping is off and `resultFormat` is set to `json` or `json/raw`, every record is printed in a separate document and statistics and warnings are printed in plain text.

  For example if MySQL Shell is started without `--json` and `resultFormat=json/raw`:

```
mysqlsh-sql> SHOW DATABASES;
{"Database":"information_schema"}
{"Database":"mysql"}
{"Database":"performance_schema"}
{"Database":"sys"}
4 rows in set (0.0035 sec)
```

  If MySQL Shell is started with `--json` and with `resultFormat=json/raw`:

```
mysqlsh-sql> SHOW DATABASES;
{
    "hasData": true,
    "rows": [
        {
            "Database": "information_schema"
        },
        {
            "Database": "mysql"
        },
        {
            "Database": "performance_schema"
        },
        {
            "Database": "sys"
        }
    ],
    "executionTime": "0.0018 sec",
    "affectedRowCount": 0,
    "affectedItemsCount": 0,
    "warningCount": 0,
    "warningsCount": 0,
```

```
    "warnings": [],
    "info": "",
    "autoIncrementValue": 0
}
```

Bugs Fixed

- MySQL Shell could be installed in an environment where Python was not present, but the application has a dependency on many standard Python modules, resulting in error messages at startup. The RPM and Debian packages for MySQL Shell now explicitly specify the dependency on Python. (Bug #29469201)
- The MSI file that is used by Windows Installer to install MySQL Shell now adds the path to the application binary (`mysqlsh`) to the Windows `PATH` environment variable, so that the application can be started from a command prompt. (Bug #29457639)
- In the instructions to build MySQL Shell from source (the `INSTALL` document), the required version of the optional V8 dependency has been updated from 3.28.71.19 to 6.7.288.46. Thanks to Evgeniy Patlan for spotting this. (Bug #29430049, Bug #94529)
- MySQL Shell's upgrade checker utility `checkForServerUpgrade()` could incorrectly report a schema inconsistency error for a table whose name included a special character such as a hyphen. (Bug #29346836, Bug #94303)
- On Windows, MySQL Shell's upgrade checker utility `checkForServerUpgrade()` incorrectly reported a schema inconsistency error for partitioned tables. (Bug #29256562)
- MySQL Shell stopped unexpectedly if Python code was running in interactive mode and threw exceptions from C++ libraries. These exceptions are now caught and translated to Python's built-in RuntimeError exceptions. (Bug #29057116)
- When a connection is specified using key-value pairs in MySQL Shell's `shell.connect()` method, the host name cannot be an empty string. MySQL Shell now handles this situation consistently and returns an error if the supplied host name is an empty string. (Bug #28899522)
- MySQL Shell's JSON import utility can now accept input from FIFO special files (named pipes) when you invoke the utility using the `util.importJSON` function, so you can carry out large imports by this method without needing to put the data into a file. (Bug #28785527)
- When you use the MySQL Shell command `\help` (or `\h`, or `\?`) with a search pattern to search for help on a specific subject, multiple help topic titles can match the pattern and be returned as a list, to be selected by entering the command again with an extended search pattern. With this system, it was possible for help topics with a single-word title to be inaccessible from such a list because there was nothing further to add to the search pattern. To avoid this situation, the handling of multiple matches has now been improved. If a topic title is found that matches the given search pattern exactly (case-sensitive in the event of multiple topic matches, and case-insensitive in the event of no case-sensitive matches), the topic is identified as the exact match and its help data is printed. The rest of the topics with pattern matches in their titles are listed in a “see also” section and can be selected by further pattern matching. (Bug #28393119)

Changes in MySQL Shell 8.0.15 (2019-02-01, General Availability)

This release contains no functional changes and is published to align version number with the MySQL Server 8.0.15 release.

Changes in MySQL Shell 8.0.14 (2019-01-21, General Availability)

- AdminAPI Added or Changed Functionality

AdminAPI Added or Changed Functionality

- The *`Cluster`*`.status()` operation has been extended to enable you to display information about the underlying Group Replication group used by the cluster. Now you can retrieve information from all members of a cluster without having to connect to each member individually.

  To see information about the `groupName` and `memberId`; and general statistics about the number of transactions checked, proposed, and rejected by members issue:

  ```
  Cluster.status({extended:true})
  ```

  To see information about recovery and regular transaction I/O, applier worker thread statistics and any lags; applier coordinator statistics, if parallel apply is enabled; error, and other information from I/O and applier threads issue:

  ```
  Cluster.status({queryMembers:true})
  ```

  In addition, in previous versions the URI-type string shown for `groupInformationSourceMember` in the output of *`Cluster`*`.status()` could be the cluster's MySQL Router address, rather than the address of the instance which provided the displayed group information. This has been improved to ensure `groupInformationSourceMember` always shows the correct `hostname`, or `report_host`, value and `port`, or `report_port`, value of the instance which provided the group information. (Bug #28636963, Bug #26519466, Bug #27824265, Bug #28366027)

- AdminAPI no longer relies on the mysqlprovision check command. This work has resulted in the following:

  - The `errors` field in the JSON returned by `dba.checkInstanceConfiguration()` has been removed, because it was only used to hold errors issued by mysqlprovision. Any errors are now reported directly, for example as RuntimeError.

  - The `dba.verbose` value no longer influences the amount of debug information displayed for `dba.checkInstanceConfiguration()`, `dba.configureInstance()`, and `dba.configureLocalInstance()` because it was only used to control the verbosity of the information displayed from mysqlprovision. Instead, the generic verbose value from MySQL Shell is used to control the verbosity level for those functions.

  - In addition, the messages returned have been generally improved to make them more accurate.

  References: See also: Bug #28737777, Bug #27305806, Bug #28768627, Bug #27702439, Bug #28733883.

- When you create a cluster, you can set the timeout before instances are expelled from the cluster, for example when they become unreachable. Pass the new `expelTimeout` option to the `dba.createCluster()` operation, which configures the `group_replication_member_expel_timeout` variable on the seed instance. All instances running MySQL server 8.0.13 and later which are added to the cluster are automatically configured to have the same `group_replication_member_expel_timeout` value as configured on the seed instance.

- You can now configure an InnoDB Cluster's mode while the cluster remains online. This enables you to configure the underlying Group Replication group to choose a specific instance as the new primary in

single-primary mode, or to change between multi-primary and single-primary modes without taking the cluster offline. This uses the group coordinator and the UDFs added in WL#10378, see Configuring an Online Group. Use the following operations:

- `Cluster.setPrimaryInstance(instance)`, which forces the election of `instance` as the new primary by overriding any election process.
- `Cluster.switchToMultiPrimaryMode()`, which switches the cluster to multi-primary mode. All instances become primaries.
- `Cluster.switchToSinglePrimaryMode([instance])`, which switches the cluster to single-primary mode. If `instance` is specified, it becomes the primary and all the other instances become secondaries. If `instance` is not specified, the new primary is the instance with the highest member weight (and the lowest UUID in case of a tie on member weight).

- You can now check and modify the settings in place for an InnoDB Cluster while the instances are online. To check the current settings of a cluster, use the following operation:

  - `Cluster.options()`, which lists the cluster configuration options for its ReplicaSets and instances. A boolean option `all` can also be specified to include information about all Group Replication system variables in the output.

  This work also enables you to configure the InnoDB Cluster options at a cluster level or instance level, while instances remain online. This avoids the need to remove, reconfigure and then again add the instance to change InnoDB Cluster options. Use the following operations:

  - `Cluster.setOption(option, value)` to change the settings of all cluster instances globally
  - `Cluster.setInstanceOption(instance, option, value)` to change the settings of individual cluster instances

  The way which you use InnoDB Cluster options with the operations listed depends on whether the option can be changed to be the same on all instances or not. These options are changeable at both the cluster (all instances) and per instance level:

  - `exitStateAction`
  - `memberWeight`

  This option is changeable at the per instance level only:

  - `label`

  These options are changeable at the cluster level only:

  - `failoverConsistency`
  - `expelTimeout`
  - `clusterName`

- The `Cluster.rescan()` operation has been extended to enable you to detect changes to the cluster's topology, and modify the cluster metadata, for example to remove old instance data. Now you can:

  - use the `updateTopologyMode` option to detect if the Group Replication mode (single-primary or multi-primary mode) registered in the metadata matches the current mode of the cluster, updating that information in the metadata if requested through a new

option or by a prompt confirmation. You can use this option to update the metadata after using the *`Cluster`*`.switchToSinglePrimaryMode([`*`instance`*`])` and *`Cluster`*`.switchToMultiPrimaryMode()` options added in WL#12052.

- use the `addInstances` option to specify a list of new instances to add to the metadata, or the `removeInstances` option to specify a list of obsolete instances to remove from the metadata. Pass the `auto` value to these options to automatically add or remove instances from the metadata, without having to specify an explicit list of instances. This enables the function to update the metadata even in noninteractive mode, making it consistent with the other AdminAPI operations.
- In addition, a new interactive option has been added to the *`cluster`*`.rescan()` operation, to enable or disable interactive mode prompts specifically for the *`cluster`*`.rescan()` command.

References: See also: Bug #28997465, Bug #28529362, Bug #28889563, Bug #25675665, Bug #28542904.

- In 8.0.14, Group Replication introduces the ability to specify the failover guarantees (eventual or “read your writes”) if a primary failover happens in single-primary mode (see WL#11123). Configure the failover guarantees of an InnoDB Cluster at creation by passing the new `failoverConsistency` option to the `dba.createCluster()` operation, which configures the `group_replication_consistency` system variable on the seed instance. This option defines the behavior of a new fencing mechanism used when a new primary is elected in a single-primary group. The fencing restricts connections from writing and reading from the new primary until it has applied any pending backlog of changes that came from the old primary (sometimes referred to as “read your writes”). While the fencing mechanism is in place, applications effectively do not see time going backward for a short period of time while any backlog is applied. This ensures that applications do not read stale information from the newly elected primary.

  The `failoverConsistency` option is only supported if the target MySQL server version is 8.0.14 or later, and instances added to a cluster which has been configured with the `failoverConsistency` option are automatically configured to have `group_replication_consistency` the same on all cluster members that have support for the option. The variable default value is controlled by Group Replication and is `EVENTUAL`, change the `failoverConsistency` option to `BEFORE_ON_PRIMARY_FAILOVER` to enable the fencing mechanism. Alternatively use `failoverConsistency=0` for `EVENTUAL` and `failoverConsistency=1` for `BEFORE_ON_PRIMARY_FAILOVER`.

  

  Note

  Using the `failoverConsistency` option on a multi-primary InnoDB Cluster has no effect but is allowed because the cluster can later be changed into single-primary mode with the *`Cluster`*`.switchToSinglePrimaryMode()` operation.

AdminAPI Bugs Fixed

- The default value for `group_replication_exit_state_action` is `ABORT_SERVER`, but AdminAPI now overrides this and sets the default on instances to `READ_ONLY`. This ensures that instances which leave the group unexpectedly continue running and can be rejoined to the cluster. (Bug #28701263)
- When a cluster was created on a server that did not have the X Plugin enabled, a silent assumption was being made about the X Protocol port value. Now the value of an X Protocol port is only stored for instances on which X Plugin is enabled. (Bug #27677227)
- The `dba.checkInstanceConfiguration()` operation was not checking if the Performance Schema was enabled on the target instance. This could result in a situation where you could create a cluster but could not run several management operations on it, for example the *`Cluster`*`.status()` operation.

Now, `dba.checkInstanceConfiguration()` checks that the Performance Schema is enabled on instances. (Bug #25867733)

- When `Cluster.checkInstanceState()` was executed on an instance which was already a member of the current cluster, the output indicated that the instance was fully recoverable. This was misleading and was caused by a missing validation to ensure the instance does not belong to a cluster. (Bug #24942875)

- The `dba.checkInstanceConfiguration()` operation did not recognize privileges when they were associated to a user through a role (available in MySQL server 8.0 and higher). In such a case, a missing privileges error was being incorrectly issued despite the user possessing all the required privileges. Now users with their privileges assigned by roles are recognized by AdminAPI operations correctly. (Bug #91394, Bug #28236922)

Functionality Added or Changed

- **X DevAPI:** The `Table` and `Collection` objects now support the `.count()` method, part of the X DevAPI.

- When started from the command line, MySQL Shell prints information about the product, information about the session (such as the default schema and connection ID), warning messages, and any errors that are returned during startup and connection. You can now suppress printing of information that you do not need by using the `--quiet-start[=1|2]` `mysqlsh` command-line option. With a value of 1 (the default when the option is specified), information about the MySQL Shell product is not printed, but session information, warnings, and errors are printed. With a value of 2, only errors are printed.

  As part of this work, the printed information was tidied up so that the information about the MySQL Shell product is printed before the information about the session. Also, the handling of error printing was normalized to send diagnostic data to `stderr`, and errors to `stdout`. (Bug #28833718, Bug #28855291)

- MySQL Shell connections using classic MySQL protocol now support compression for information sent between the client and the server. You can specify compression when you start MySQL Shell and connect using command line options, or in a URI string or a key-value pair when you create a session using other interfaces. You can also use the MySQL Shell configuration option `defaultCompress` to enable compression for every global session.

  For MySQL Shell connections that use Unix socket files, the `--socket` command line option can now be specified with no argument to connect using the default Unix socket file for the protocol. (Bug #28730149)

- The MySQL Shell JSON import utility can now process BSON (binary JSON) data types that are represented in JSON documents. The data types used in BSON documents are not all natively supported by JSON, but can be represented using extensions to the JSON format. The import utility can process documents that use JSON extensions to represent BSON data types, convert them to an identical or compatible MySQL representation, and import the data value using that representation. The resulting converted data values can be used in expressions and indexes, and manipulated by SQL statements and X DevAPI functions.

  To convert JSON extensions for BSON types into MySQL types in this way, you must specify the `convertBsonTypes` option when you run the import utility. Additional options are available to control the mapping and conversion for specific BSON data types. If you import documents with JSON extensions for BSON types and do not use this option, the documents are imported in the same way as they are represented in the input file.

- A MySQL Shell configuration option `showColumnTypeInfo` and command line option `--column-type-info` have been added to display metadata for each column in a returned result set, such as the column type and collation. The metadata is printed before the result set, and is only shown in SQL mode.

  In the metadata, the column type is returned as both the type used by MySQL Shell (`Type`), and the type used by the original database (`DBType`). For MySQL Shell connections using classic MySQL protocol, `DBType` is as returned by the protocol, and for X Protocol connections, `DBType` is inferred from the available information. The column length (`Length`) is returned in bytes.

- The upgrade checker utility provided by MySQL Shell, which is the `checkForServerUpgrade()` function of the `util` global object, has several enhancements:

  - The utility can now select and provide advice and instructions for relevant checks that cannot be automated, and must be performed manually. The manual checks are rated as either warning or notice (informational) level, and are listed after the automated checks. In MySQL Shell 8.0.14, the utility provides advice where relevant about the change of default authentication plugin in MySQL 8.0.

  - A check has been added for the removed `log_syslog_*` system variables that previously configured error logging to the system log (the Event Log on Windows, and `syslog` on Unix and Unix-like systems).

  - A check has been added for specific schema inconsistencies that can be caused by the deletion or corruption of a file, including the removal of the directory for a schema and the removal of a `.frm` file for a table.

  You can access the upgrade checker utility from within MySQL Shell or start it from the command line. For instructions and further information, see MySQL Shell Utilities.

- MySQL Shell can print results in table, tabbed, or vertical format, or as pretty or raw JSON output. From MySQL Shell 8.0.14, the new MySQL Shell configuration option `resultFormat` can be used to specify any of these output formats as a persistent default for all sessions, or just for the current session. Changing this option takes effect immediately. Alternatively, the new command line option `--result-format` can be used at startup to specify the output format for a session. The existing command line options `--table`, `--tabbed`, and `--vertical` are now aliases for the `--result-format` option given with the corresponding value.

  The existing command line option `--json` controls JSON wrapping for all MySQL Shell output from a session. Specifying `--json` or `--json=pretty` turns on JSON wrapping and generates pretty-printed JSON. Specifying `--json=raw` turns on JSON wrapping and generates raw JSON. With any of these options, the value of the `resultFormat` MySQL Shell configuration option is ignored. Specifying `--json=off` or not specifying the `--json` option turns off JSON wrapping, and result sets are output as normal in the format specified by the `resultFormat` configuration option.

  The `outputFormat` MySQL Shell configuration option is now deprecated. This option combined the JSON wrapping and result printing functions, which have now been separated. If this option is still specified in your MySQL Shell configuration file or scripts, the behavior is as follows:

  - With the `json` or `json/raw` value, `outputFormat` activates JSON wrapping with pretty or raw JSON respectively.

  - With the `table`, `tabbed`, or `vertical` value, `outputFormat` turns off JSON wrapping and sets the `resultFormat` MySQL Shell configuration option for the session to the appropriate value.

- The V8 library used by MySQL Shell has been updated to version 6.7.288.46.

Bugs Fixed

- The TAR build of MySQL Shell comes with Python 2.7. When attempting to include the site package, an error was emitted because of missing build files needed by the include. (Bug #28973138)
- Handling procedures for user-supplied data in MySQL Shell were refactored to ensure correct cleanup after use. (Bug #28915716)
- The exception type and error messages returned by MySQL Shell functions for parameter errors have been standardized across the different functions. (Bug #28838958)
- MySQL Shell stopped unexpectedly if the `shell.setCurrentSchema()` method was called to set the default schema before an active session had been established. MySQL Shell now validates that there is an active session when the operation takes place. (Bug #28814112)
- The MySQL Shell JSON import utility no longer requires an empty dictionary to be supplied if there are no import options. (Bug #28768585)
- In SQL mode, MySQL Shell does not add statements to the history if they include the strings `IDENTIFIED` or `PASSWORD`, or other strings that you configure using the `--histignore` command option or `shell.options["history.sql.ignorePattern"]`. However, this previously meant that filtered-out statements were not available to be corrected immediately after entry, and had to be re-typed in case of any errors. MySQL Shell now always makes the last executed statement available to be recalled by pressing the Up arrow, regardless of the filters set in the history ignore list. If filtering applies to the last executed statement, it is removed from the history as soon as another statement is entered, or if you exit MySQL Shell immediately after executing the statement. (Bug #28749037)
- The result printing logic in MySQL Shell has been refactored to use back-end rather than high-level result data, delivering performance improvements for all types of result data and more accurate representation for JSON data. (Bug #28710831)
- A memory leak was fixed that occurred when the new MySQL Shell command-line syntax was used. (Bug #28705373)
- The check for partitioned tables in shared tablespaces in the upgrade checker utility provided by MySQL Shell (the `util.checkForServerUpgrade()` operation) did not return correct results for the 8.0.11 and 8.0.12 target versions. The check now uses alternative Information Schema tables that are populated with the required information in these versions. (Bug #28701423)
- The MySQL Shell command `\option` ignored additional arguments separated by spaces that were specified for an option after the initial value. (Bug #28658632)
- MySQL Shell permitted newline characters (line feed and carriage return) in passwords to be passed to a Secret Store Helper using the `shell.storeCredential` method, resulting in an error in the Secret Store Helper. MySQL Shell now returns an exception if newline characters are used in supplied passwords for the `shell.storeCredential` method, and does not pass them to the Secret Store Helper. (Bug #28597766)
- On the Windows platform, UTF-8 encoded strings were printed to the console using the `cout` object, which transfers a byte at a time. This resulted in multi-byte Unicode characters, such as a single quotation mark, being displayed and handled incorrectly. MySQL Shell now uses alternative functions for printing, and verifies that multi-byte UTF-8 characters are emitted as a complete unit. (Bug #28596692)
- When executing an SQL script in MySQL Shell, an inaccurate line number was reported for the location of syntax errors in the script. The number referenced the current SQL block rather than the line number in the script. The error message now uses the global line number. (Bug #28545982)

- The SQL statement splitting logic in MySQL Shell has been refactored to fix a number of issues and to match behaviors of the MySQL command-line tool `mysql`:

  - The backslash character (`\`) is no longer accepted in the delimiter string.
  - The use of the word “delimiter” in contexts other than as a command is now handled correctly.
  - In scripts, comments are not discarded, and groups of comments and statements are now split in the same way as `mysql` would split them.
  - Large scripts can now be successfully split into incremental chunks even when some tokens span across more than one chunk.
  - Scripts can now be parsed in the `ANSI_QUOTES` SQL mode.
  - Multi-line strings and comments that contain quotes are now parsed correctly.
  - Inline commands are handled in the same way as by `mysql`, as follows:

    - A `\` character appearing at the beginning of a statement is interpreted as the start of a multi-letter MySQL Shell command.
    - A `\` character appearing within a statement is interpreted as the start of a single-letter command. The command is executed immediately, then stripped out of the input statement.
    - A `\` character appearing after the end of a statement is interpreted as the start of a single-letter command.

  (Bug #27959016, Bug #25689071)

- The handling of Windows named pipe connections by MySQL Shell has been improved and systematized. Now, if you specify the host name as a period (.) on Windows, MySQL Shell connects using a named pipe.

  - If you are connecting using a URI type string, specify `user@.`
  - If you are connecting using a data dictionary, specify `{"host": "."}`
  - If you are connecting using individual parameters, specify `--host=.` or `-h .`

  By default, the pipe name `MySQL` is used. You can specify an alternative named pipe using the `--socket` option or as part of the URI type string. If a URI type string is used, the named pipe must be prepended with the characters `\\.\` as well as being either encoded using percent encoding or surrounded with parentheses, as shown in the following examples:

  ```
  (\\.\named:pipe)
  \\.\named%3Apipe
  ```

  (Bug #27381738)

- When JSON format output was enabled for MySQL Shell, the properties of the Shell API Options class (`shell.options`) and AdminAPI Cluster class (`dba.getCluster`) were not printed, only the class name. (Bug #25027181)

Changes in MySQL Shell 8.0.13 (2018-10-22, General Availability)

- AdminAPI Added or Changed Functionality

AdminAPI Added or Changed Functionality

- The behavior of *`Cluster`*`.dissolve()` has been updated to make it more consistent with other AdminAPI commands. Now you do not have to pass in the `force` option to start the command, and there is an interactive prompt available. When all instances belonging to the cluster are online, if MySQL Shell is running in interactive mode then you are prompted to confirm the operation of dissolving the cluster. If MySQL Shell is running in noninteractive mode, when all instances are reachable, or online, then the command removes the instances from the cluster. In the case that instances are not reachable an error is thrown. Pass in the `force` option to remove instances which are not reachable.

  References: See also: Bug #27833605, Bug #27837231.

- A new optional parameter `exitStateAction` can be used with the *`dba.`*`createCluster()` and *`cluster.`*`addInstance()` commands, which enables you to configure the `group_replication_exit_state_action` variable of an InnoDB Cluster member. The `group_replication_exit_state_action` variable enables you to specify what action is taken if a member involuntarily leaves the group. When `group_replication_exit_state_action` is set to `ABORT_SERVER` (the default value), the instance shuts itself down, and when `group_replication_exit_state_action` is set to `READ_ONLY` the instance switches itself to super read only mode instead and goes into the Group Replication `ERROR` state.

- The new optional `memberWeight` option can be used with the `dba.createCluster()` and *`Cluster.`*`addInstance()` functions to enable you to set the `group_replication_member_weight` system variable of an InnoDB Cluster server instances in a single-primary cluster. The default value is 50, in other words the system variable default. Set the `memberWeight` option to an integer between 0 and 100 to configure a member's weight in the failover election process. The value determines the chance of the instance being elected as the primary in the event of a failover. See Single-Primary Mode for more information.

AdminAPI Bugs Fixed

- The `dba.deploySandboxInstance()` function in version 8.0.12 deploys the sandbox and includes `log_syslog=OFF` in the instance's configuration file. This variable was deprecated in MySQL 8.0.12 and was removed in MySQL 8.0.13. Now, the variable has been updated to include the `loose_` prefix which makes the server ignore it for a MySQL 8.0.13 sandbox, while maintaining compatibility with earlier version sandboxes. (Bug #28543536)

- Occasionally, when adding an instance to an existing cluster the instance got stuck in the distributed recovery phase resulting in an immutable reported status of `RECOVERING`. This issue was related to the automatically generated password for the internal replication users created by InnoDB Cluster. (Bug #28219398, Bug #91348)

- In the default interactive mode, whenever using the function `dba.rebootClusterFromCompleteOutage()` without any parameter, the function failed with an error specifying the cluster name does not exist. Now the default cluster is assumed when the function is issued without a parameter. (Bug #28207565)

- The handling of metadata server changes related to the *`Cluster`*`.addInstance()` has been improved, resulting in the following changes:

  - the correct session is now used for metadata and group operations
  - the `Cluster.addInstance()` operation aborts and recommends the use of `Cluster.rescan()` if the instance is in the group but not the InnoDB Cluster metadata
  - the unnecessary parameter `super_user_password` has been removed

  (Bug #28200661)

- The Windows scripts generated by `dba.deploySandboxInstance()` incorrectly displayed user output messages in quotes. Additionally, the scripts have been improved and they no longer display executed commands. (Bug #28199954)
- The `dba.createCluster()` AdminAPI operation always created replication users, even when the `adoptFromGR` option was used. However, when adopting an already existing Group Replication group no additional users need to be created. (Bug #28054500)
- Error messages issued by the AdminAPI relating to an invalid number of arguments for a function did not include the relevant object and method prior to the message text. These messages have now been standardized. (Bug #27832594)
- When using `dba.createCluster()` or `Cluster.addInstance()`, the AdminAPI was setting the values of `auto_increment_offset` and `auto_increment_increment` incorrectly. Now the variables are set according to the following logic:
  - for a cluster running in single-primary mode:
    - `auto_increment_offset=2`
    - `auto_increment_increment=1`
  - for a cluster running in multi-primary mode:
    - `auto_increment_offset=1` and `server_id` % 7
    - `auto_increment_increment=7`

  (Bug #27084767)
- AdminAPI was using the incorrect terms for Group Replication. Now clusters are described as single-primary and multi-primary, the `multiMasterp` option has been deprecated, and the `multiPrimary` option has been added. (Bug #25926603)
- The result of calling `dba.get_cluster().status()` when quorum was lost could not be converted to a JSON object, because the string representation of the resultant object contained escape sequences. This issue was not limited to the `Cluster.status()` method, but affected all arrays and dictionaries returned by the Shell API in Python mode. The internal representation of arrays and dictionaries has been fixed. (Bug #91304, Bug #28200499)

Functionality Added or Changed

- **X DevAPI:** The `connect-timeout` connection path parameter (see Connecting to the Server Using URI-Like Strings or Key-Value Pairs) has been added to the X DevAPI, which enables you to specify the number of seconds clients such as MySQL Shell wait until the client stops trying to connect to an unresponsive MySQL server. The value of `connect-timeout` must be a non-negative integer that defines a time frame in milliseconds. The timeout default value is 10000 milliseconds, or 10 seconds. For example:

```
// Decrease the timeout to 2 seconds.
mysqlx.getSession('user@example.com?connect-timeout=2000');

// Increase the timeout to 20 seconds
mysqlx.getSession('user@example.com?connect-timeout=20000');
```

To disable the timeout set the value to 0, meaning that the client waits until the underlying socket times out, which is platform dependent.

- The upgrade checker utility provided by MySQL Shell, which is the `checkForServerUpgrade()` function of the `util` global object, has several enhancements:
  - You can now use the upgrade checker utility to check servers at earlier MySQL 8.0.x releases, as well as MySQL 5.7 servers, for compatibility errors and issues for upgrading.
  - You can now specify a target MySQL Server version to which you plan to upgrade. In MySQL Shell 8.0.13, you can specify release 8.0.11 (the MySQL Server 8.0 GA release), 8.0.12, or 8.0.13. The upgrade checker utility carries out the checks that are relevant for the specified target release. If you specify the short form version number 8.0, or omit the `targetVersion` option, the utility checks for upgrade to the MySQL Server release number that matches the current MySQL Shell release number, currently 8.0.13.
  - A check has been added for the removed syntax `GROUP BY ASC/DESC`, returning an error message if this syntax is found in a trigger, event, view, stored procedure, or function.
  - A check has been added for columns defined as `ENUM` or `SET` that contain elements longer than 255 characters, returning an error message if any such columns are found.
  - The upgrade checker utility no longer returns a value, making its output easier to parse and process when automation is used.

  You can access the upgrade checker utility from within MySQL Shell or start it from the command line. For instructions and further information, see MySQL Shell Utilities.
- MySQL Shell onscreen output can now be displayed using a pager such as `less` or `more`. You can configure the pager you want to use with the `shell.options[`*`pager`*`]` option, the `\pager` command, or the `--pager` command option. This improves how you work with longer text output in MySQL Shell, specifically the online help and the results of SQL operations. See Using a Pager.
- The integration of MySQL Shell into command-line environments has been improved. Use the `mysqlsh` *`[options] -- shell_object object_method [method_arguments]`* syntax to pass operations directly to MySQL Shell global objects, bypassing the REPL interface. For example:

```
mysqlsh -- util check-for-server-upgrade user@example --output-format=JSON
```

  which executes the equivalent `util.checkForServerUpgrade(user@example, {"outputFormat": "JSON"})` with MySQL Shell and returns the output in JSON format. This makes it easy to integrate MySQL Shell into your automation scripts. To get help for this interface, use the MySQL Shell command `\help cmdline`. See API Command Line Interface.
- MySQL Shell has a new JSON import utility that enables you to import JSON documents from a file or standard input to a MySQL Server collection or relational table. The utility parses and validates the supplied JSON documents automatically and inserts them into the target database, removing the need to use multiple INSERT statements or write scripts to achieve this task. The utility can be started in a MySQL Shell session with the `util.importJson()` method in JavaScript or the `util.import_json()` method in Python. From the command line, you can use the `-- util importJson` syntax or the `--import` command to invoke the utility.

Bugs Fixed

- The upgrade checker utility provided by MySQL Shell (the `util.checkForServerUpgrade()` operation) did not report removed functions if they were used in views or events. (Bug #28642534)
- MySQL Shell incorrectly labeled warning messages as error messages in JSON output. (Bug #28546510)
- When MySQL Shell server connection passwords are persisted using a Secret Store, if a classic MySQL protocol connection was made without specifying a port or socket, the saved password could not be retrieved for a subsequent connection. The password storage and retrieval process now ensures that the server URL used to store the password matches subsequent queries with user-provided connection options, even if defaults were used for the original connection. (Bug #28544628)
- A number of improvements have been made to the MySQL Shell prompt, including handling of overlength text, statement splitting, and support for multiple-line prompts. New sample prompt theme files are provided for double-line prompts that use one line for information display and a new line for the input prompt itself, so that additional information can be shown without detracting from the space available for text entry. (Bug #28515394, Bug #92048)
- The `--table` command line option did not produce the appropriate output format in noninteractive mode. (Bug #28511408)
- Extra spaces before or after the parameter used with the `\help` command are now trimmed. Previously, the presence of extra spaces made MySQL Shell unable to find the relevant help topic. (Bug #28508724, Bug #92030)
- Some native MySQL Shell objects were not properly wrapped into JavaScript objects, causing memory leaks. The memory-handling mechanism has been corrected. (Bug #28473341)
- If an empty string was provided as the argument for a MySQL Shell command-line option that expects a non-null argument and has no default defined, MySQL Shell did not return an appropriate error. The error handling for command-line arguments has now been improved so that a suitable error is issued in this situation and execution of the command terminates. (Bug #28378553)
- If a session was explicitly closed by the user without closing MySQL Shell, the prompt continued to display the details of the closed session. (Bug #28314383)
- User credentials stored by MySQL Shell could not be automatically retrieved for hosts identified by IPv6 addresses. (Bug #28261301)
- MySQL Shell now displays the build type (commercial or Community Edition) as part of the product version information displayed at startup and when the `--help` argument is used. (Bug #28242573)
- If a MySQL Shell session was disconnected without closing MySQL Shell (for example, using the `session.close()` method), a subsequent query in SQL mode did not return a "Not connected" error. MySQL Shell now checks not only that the global session object exists, but also that it has a valid connection to the MySQL server instance. (Bug #28240437)
- On the Windows platform, the background color was not reset in the MySQL Shell window when the terminal was cleared using **Ctrl+L**. (Bug #28235701, Bug #91102)
- The commercial MySQL Shell package could not be installed on Debian or Ubuntu if the equivalent Community Edition package had already been installed. (Bug #28223781)
- MySQL Shell was using some deprecated functions and properties internally, which caused warning messages to appear in the MySQL Shell log file, although the functions were not executed by users. The deprecated functions and properties have now been removed from the internal code. (Bug #28216558)

- If an invalid value was specified for the MySQL Shell option `credentialStore.helper`, the resulting error message at MySQL Shell startup was displayed incorrectly. (Bug #28216485)
- The upgrade checker utility provided by MySQL Shell (the `util.checkForServerUpgrade()` operation) now correctly handles account names with spaces and other blank characters, and skips the permissions check when the server was started with the `--skip-grants` option. (Bug #28212899, Bug #91326)
- When deleting history entries using the `\history` command in MySQL Shell, you can now specify a number of history entries to be deleted from the tail of the history, using the format `\history delete -number`. The handling of history entry numbers for deleted entries has been improved so that when the tail of the history is deleted, those history entry numbers are reused for new entries, and there is no gap. If a `\history delete` command empties the history, the numbering of history entries now resets as it does when the `\history clear` command is used. Also, the error message issued if you specify an invalid range of history entries to be deleted (using the format `\history delete firstnumber-lastnumber`) has been improved. (Bug #28199513)
- On the Windows platform, using the **Ctrl+C** key combination in MySQL Shell caused MySQL Shell to close, even if a command was in progress. (Bug #27894642)
- The MySQL Shell code for identifying whether a given IP address is a loopback address did not account for additional IP addresses (besides the 127.0.0.0/8 address block) that had been added by the user to the loopback interface. All IP addresses assigned to the loopback interface are now checked. (Bug #27703779)
- When running MySQL Shell in batch mode, tab separated format is the default for output. In some situations, table format was used for output instead. This issue has now been fixed. A command line option `--tabbed` has also been added to switch to the tab separated format for output when MySQL Shell is in interactive mode, where the default is table format. (Bug #27546082, Bug #89514)
- Zone IDs in IPv6 addresses are now supported for MySQL Shell connections. A zone identifier is suffixed to the IPv6 address with a percent character (%) as a separator. For example:

  ```
  2001:0db8:3c4d:0015::1a2f:1a2b%14
  ```

  If an IPv6 address with a zone ID is provided as a URI type string, URL encoding (percent-encoding) must be used for the percent character. For example:

  ```
  mysqlsh --uri=user@[2001:0db8:3c4d:0015::1a2f:1a2b%2514]:33060
      shell.connect("user@[2001:0db8:3c4d:0015::1a2f:1a2b%2514]:33060")
  ```

  If an IPv6 address with a zone ID is provided using individual parameters or a data dictionary, URL encoding does not need to be used for the percent character, and the IPv6 address can be supplied as seen. For example:

  ```
  mysqlsh --user=user --host=2001:0db8:3c4d:0015::1a2f:1a2b%14 --port=33060
  ```

  See Connecting to the Server Using URI-Like Strings or Key-Value Pairs. (Bug #27539702)
- When MySQL Shell log entries were output to `stderr` by prepending @ (at sign) to the value of the `--log-level` option, and the JSON output format was selected for MySQL Shell, some log entries were not being output in JSON format. The logger now checks and uses the current value of the MySQL Shell `outputFormat` configuration option as the output format when writing log entries to `stderr`. (Bug #27480887)
- In SQL mode, MySQL Shell erroneously entered multi-line mode if an unknown command was executed, or if multiple consecutive SQL statement delimiters were used. Now, an appropriate error is returned in these situations. (Bug #27411526)

- If you do not specify a protocol with the `\connect` command or when starting MySQL Shell, MySQL Shell automatically attempts to use X Protocol for the session's connection, and falls back to classic MySQL protocol if X Protocol is unavailable. The connection type option `-ma`, which specified that behavior explicitly, is now deprecated.

  The use of a single dash with the connection type options `-mx` and `-mc`, for an X Protocol and classic MySQL protocol connection respectively, is also deprecated. These options must now be specified with a double dash (that is, `--mx` and `--mc`) with the `\connect` command or when starting MySQL Shell. They are now defined as aliases of the long form `--mysql` (`--mc`) and `--mysqlx` (`--mx`) connection type options. (Bug #27363459)

- On the Windows platform, when a long command was accessed from the MySQL Shell command history and edited at the right edge of the console window, a cursor positioning error caused the command to move up one line and overwrite the output of previous commands. The issue has now been fixed. (Bug #27068352)

- The mysqlsh command-line options `--dbpassword[=password]` and `--dbuser=user_name` are now deprecated. Use the options `--password` (`-p`) and `--user` (`-u`) instead. (Bug #26049681)

- The result printer in MySQL Shell was refactored to improve handling of binary data based on the output format, handling of multi-byte characters, and alignment of table formatting when multi-line characters are present. (Bug #24912154, Bug #24967872)

Changes in MySQL Shell 8.0.12 (2018-07-27, General Availability)

- AdminAPI Added or Changed Functionality
- AdminAPI Bugs Fixed
- Functionality Added or Changed
- Bugs Fixed

AdminAPI Added or Changed Functionality

- The *`Cluster`*`.removeInstance()` command has been improved, with the following changes:

  - A new interactive option has been added to enable or disable interactive mode for the command. The output displayed in interactive mode has been improved, displaying more useful information.In interactive mode, you are prompted to continue with the removal of the instance (or not) in case it is not reachable.

  - The operation now ensures that the instance is removed from the metadata of all the cluster members and itself. This only applies to `ONLINE` members.

  - A new global option `dba.gtidWaitTimeout` is available to define the timeout to wait for transactions (GTIDs) to be applied when required by AdminAPI commands. If the timeout value defined by `dba.gtidWaitTimeout` is reached when waiting for the cluster transactions to be applied for *`cluster`*`.removeInstance()` and `force: false` (or not defined) then an error is issued and the operation aborted. When `force: true` then the operation continues and does not generate an error.

  References: See also: Bug #27817894.

- When using the `ipWhitelist` to define which servers could access the cluster, the internal user accounts were not matching the whitelist. Now AdminAPI applies the same filtering logic from `ipWhitelist` for the internal administrative accounts.

References: See also: Bug #26140094, Bug #28165891.

AdminAPI Bugs Fixed

- The `cluster.forceQuorumUsingPartitionOf()` operation sets the `group_replication_force_members` variable on the target instance to force a new group membership and restore the quorum, but it did not reset the value of the variable at the end of the process. Consequently, if Group Replication later needed to be restarted on the target instance it failed because the `group_replication_force_members` variable was still set. Now, the `group_replication_force_members` variable is reset to an empty string at the end of the `cluster.forceQuorumUsingPartitionOf()` operation. (Bug #28064621)
- Some messages displayed by MySQL Shell were showing a MySQL server version that does not exist. (Bug #27924694)
- It was possible to use AdminAPI operations on server instances running an incompatible version of MySQL. (Bug #27765769)
- The setting of the `bind_address` variable is no longer a requirement. (Bug #27765484)
- When creating a cluster or adding an instance, if the `localAddress` option is not specified, the port used for `group_replication_local_address` is automatically assigned with the value: `port` * 10 + 1. However, if the resulting port determined by the previous rule was already in use then a random port was generated and used. Now MySQL Shell checks that the `group_replication_local_address` port is available, and fails if it is not. (Bug #27758041)
- The `dbPassword` option is no longer valid in the options dictionary of all AdminAPI commands. (Bug #27745106)
- It was possible to use the `dba.forceQuorumUsingPartition()` operation on a cluster which had not lost quorum. (Bug #27508698)
- The help message for `dba.rebootClusterFromCompleteOutage()` operation was incorrectly suggesting to use `dba.forceQuorumUsingPartition()`. (Bug #27508627)
- The `dba.rebootClusterFromCompleteOutage()` operation was creating a new user on the target instances, which could lead to the existence of an increasing number of users. The fix ensures that these users are not created by the `dba.rebootClusterFromCompleteOutage()` operation. (Bug #27344040)
- Now when you issue `dba.getCluster()` and retrieve a cluster without quorum a warning is issued in addition to the log message. (Bug #27148943)
- The `memberSslMode` option could be used with *`cluster`*`.addInstance()` and *`cluster`*`.rejoinInstance()` operations but if you specified a different value than the one used at cluster creation an error was thrown. Now set the SSL mode at the cluster level only, in other words when issuing `dba.createCluster()`. The `memberSslMode` option has been removed from *`cluster`*`.addInstance()` and *`cluster`*`.rejoinInstance()`. (Bug #27062122)
- When you issued `dba.configureLocalInsance()` on an instance, it configured the `disabled_storage_engines` variable with the `MyISAM`, `BLACKHOLE`, `FEDERATED`, `CSV`, and `ARCHIVE` storage engines to ensure that the storage engine was set to `InnoDB`, as required by Group Replication. The change to this option was not being reported correctly by AdminAPI, and hence the required restart after changing the `disabled_storage_engines` variable was not clear. This change was deemed a recommendation, rather than a requirement, hence `dba.configureLocalInsance()` no longer configures `disabled_storage_engines`. (Bug #26754410)

- Creating a cluster using an account which was missing the global grant option failed with an ambiguous error message, even though `dba.checkInstanceConfiguration()` did not return any errors. Now when you create a cluster, the account being used to administer the cluster is checked to ensure that it has the global grant option. (Bug #25966235)
- MySQL Shell is able to automatically reconnect global session when running in the interactive mode, but AdminAPI methods lacked this feature. This resulted in you having to reconnect manually. Now, the AdminAPI methods which utilize the global session object have been improved in order to detect an interrupted session and trigger the reconnection mechanism. The Cluster object uses its own internal session instance, which does not support automatic reconnection. If connection to the cluster is lost, you need to manually recreate the Cluster object. (Bug #24702489)
- In the event of a whole cluster stopping unexpectedly, upon reboot the `memberSslMode` was not preserved. In a cluster where SSL had been disabled, upon issuing `dba.rebootClusterFromCompleteOutage()` this could prevent instances from rejoining the cluster. (Bug #90793, Bug #27986413)

Functionality Added or Changed

- **Important Change:** An RPM package for installing ARM 64-bit (aarch64) binaries of MySQL Shell on Oracle Linux 7 is now available in the MySQL Yum Repository and for direct download.

  Known Limitation for this ARM release: You must enable the Oracle Linux 7 Software Collections Repository (ol7_software_collections) to install this package, and must also adjust the libstdc++7 path. See Yum's Platform Specific Notes for additional details.

- **X DevAPI:** In order to be compliant with the X DevAPI specification, the following changes have been made:

  - `Collection.modify(condition).arrayDelete()` and `Collection.modify(condition).merge()` have been deprecated.
  - `Collection.find().limit(x).skip(y)` has been renamed to `Collection.find().limit(x).offset(y)`.
  - `Collection.find().limit(x).skip(y)` has been deprecated.
  - `Collection.find().limit(x).offset(y)` has been implemented.
  - `BaseResult.getAffectedItemsCount()` has been implemented.
  - `BaseResult.getWarningCount()` has been deprecated.
  - `BaseResult.getWarningsCount()` has been implemented.
  - `Result.getAffectedItemCount()` has been deprecated.
  - `SqlResult.getAffectedRowCount()` has been deprecated.
  - `SqlResult.nextDataSet()` has been renamed to `SqlResult.nextResult()`.
  - `SqlResult.nextDataSet()` has been deprecated.
  - `SqlResult.nextResult()` has been implemented.

- MySQL Shell now enables you to store user credentials in an operating system specific secret store. You can then enter a MySQL user's password during connection and store it for future connections. Currently the following secret stores are supported:

  - MySQL login-path
  - MacOS keychain
  - Windows API

  (Bug #23304789, Bug #81484)

- The way you access the online Shell help has been standardized. Use the `\help` *`pattern`* command to search the help. The scope of the command has been increased to support retrieving help for the following categories:

  - Class and function help for the Admin API, X DevAPI and Shell API. Previously, to retrieve help for API objects, you had to create an instance of the object and use the *`object`*`.help()` method.
  - SQL syntax help, provided that a global session object exists.

  Wildcards can now be used to search for help. A number of additional bugs relating to incomplete help information have also been fixed. (Bug #23255291, Bug #81277, Bug #24963435, Bug #25732663, Bug #85481, Bug #25739522, Bug #85511, Bug #25739664, Bug #85514, Bug #26393155, Bug #86950, Bug #24943074, Bug #26429399, Bug #87037, Bug #27870491, Bug #90455, Bug #27870503, Bug #90456, Bug #27875150, Bug #90474, Bug #24948933, Bug #83527)

- The `util.checkForServerUpgrade()` operation has an additional `outputFormat` parameter that you can specify when running the utility. The utility can now generate output in two formats:

  - TEXT format, which is the default. This option provides output suitable for humans, as previously returned by the utility.
  - JSON format. This option provides output suitable for machines, which can be parsed and processed for various further use cases.

Bugs Fixed

- The sample prompt theme files for MySQL Shell were deployed to an incorrect location on the Windows platform, in the root install folder. The files are now correctly deployed in the `\share\mysqlsh\prompt` sub-folder. (Bug #28188761)
- The upgrade checker utility provided by MySQL Shell (the `util.checkForServerUpgrade()` operation) has been enhanced with a summary count of the errors, warnings, and information level issues found by the tool, and with links to documentation with further information where this is available. (Bug #28171814)
- When upgrading from version 1.0.11 to version 8.0.11 of MySQL Shell on Linux, the upgrade failed if the original package was the community edition and the new package was the commercial edition, or vice versa. Upgrading from one edition to the other edition is now enabled. (Bug #28037407)
- The `util.checkForServerUpgrade()` operation can now use either an X Protocol connection or a classic MySQL protocol connection. (Bug #28027707)
- The `checkForServerUpgrade()` operation to verify upgrade prerequisites included an unnecessary check relating to `ZEROFILL` and display length attributes in columns. The check has now been removed. (Bug #27927641, Bug #90634)

- For sessions using the classic MySQL protocol, if the `session_track_gtids` system variable is set on the server to capture and return GTIDs to the client, MySQL Shell now displays the GTIDs for successfully committed transactions. The returned GTID values are also now recorded in tracing information. (Bug #27871148)
- When the `defaultMode` MySQL Shell configuration option had been set with the `--persist` option, batch code execution from a file was always attempted using the specified default language, even if the file extension indicated a different supported language. Now when a file is loaded for batch processing using the `--file` or `-f` option, files with the extensions `.js`, `.py`, and `.sql` are processed in the appropriate language mode, regardless of the set default language. (Bug #27861407)
- The methods provided in the `shell.options` configuration interface to set and save persistent option values used underscores in JavaScript as well as in Python mode. The methods have now been changed to `shell.options.setPersist()` and `shell.options.unsetPersist()` in JavaScript to follow the appropriate naming convention. (Bug #27861141)
- When executing a SQL script using MySQL Shell, delimiters (such as the default semi-colon character) present in multi-line comments caused execution to fail. Delimiters are now ignored inside multi-line comments. (Bug #27841719)
- MySQL Shell returned an error when querying timestamp values that were zero, because a zero value for the month or day in a date was not accepted. Zero timestamp values can now be used without producing an error. (Bug #27833822, Bug #90355)
- The `shell.getSession()` function returns a reference to the `session` global object representing the already established connection between MySQL Shell and a MySQL server, known as a global session. MySQL Shell now gracefully handles the situation where the function is called when no global session has yet been established. (Bug #27809310)
- The MySQL Shell application icon on Microsoft Windows was not being displayed for the MySQL Shell 8.0 GA release, due to an incorrect association introduced for the icon during code refactoring. The icon is now displayed correctly. (Bug #27746532)
- The `\status` (`\s`) command in MySQL Shell now displays full information about the version and build of the connected MySQL server. (Bug #27740420)
- The check for reserved keywords carried out by the `util.checkForServerUpgrade()` operation was updated to match the list of reserved keywords for the MySQL 8.0 GA release. (Bug #27724201)
- When handling escape sequences, MySQL Shell now identifies and skips over SQL comments and string literals within quotation marks. (Bug #27665229)
- Python's mapping type has been added to MySQL Shell, so that dictionary syntax can be used to interact with data in Python mode. (Bug #27614110)
- When a file was redirected to standard input for execution in MySQL Shell, on Unix, the first part of the file was taken as being the password. The password prompt now looks for user input first before resorting to standard input. (Bug #27572380)
- If **Ctrl+C** was entered or an unexpected error occurred at a password prompt in MySQL Shell, the terminal state was not restored correctly afterwards. (Bug #27379834)

Changes in MySQL Shell 8.0.11 (2018-04-19, General Availability)

- AdminAPI Added or Changed Functionality
- AdminAPI Bugs Fixed

- Functionality Added or Changed
- Bugs Fixed

AdminAPI Added or Changed Functionality

- InnoDB Cluster instances running MySQL 8.0.11 and higher can now be configured for InnoDB Cluster usage remotely, without the need to log in to the instance and run `dba.configureLocalInstance()` locally. Use the new `dba.configureInstance()` operation, which enables remote automatic configuration of compatible instances for InnoDB Cluster usage and if necessary also supports remote restart of compatible instances after configuration.

  Similarly, compatible instances support having any changes to their settings automatically persisted by AdminAPI after any cluster topology changes. Now, when working with remote instances if you create a cluster, add an instance to a cluster, or remove an instance from a cluster supporting this functionality, there is no need to log in to the instance and run `dba.configureLocalInstance()` to persist the changes to the instance's option file. This makes working with production clusters, which are built on networked instances, much easier. This also ensures that instances automatically rejoin a cluster in the case of a restart.

  As part of this feature the following improvements were made:

  - Improved display of messages.
  - Improved detection of loopback hostnames on Debian and Ubuntu, and local instances based on the hostname.
  - Improved display of required configuration changes.
  - Replaced automatic configuration of instances after issuing `dba.createCluster` and `dba.addInstance` with the configuration check `dba.checkInstance()`, aborting the command if the check fails.
  - When specifying an account in interactive mode, if the user's hostname is local, for example localhost, you are given the option to recreate the account configured for remote usage by using the hostname `%`, or to create a new InnoDB Cluster administrator account.

  References: See also: Bug #27608299, Bug #27112727, Bug #27629803.

AdminAPI Bugs Fixed

- After creating a cluster administrative user using `dba.createCluster()`, MySQL Shell attempted to reconnect using the new user but still using localhost, which failed causing the whole configuration to fail. Now MySQL Shell no longer switches accounts between administrative account creation and instance configuration. (Bug #27673816)
- The changes introduced by Bug#27545850 mean that now it is not possible to modify Group Replication related options while the plugin is starting or stopping. The AdminAPI operations have been modified to ensure that they respect this change, and attempting to issue a statement which would change a Group Replication operation while the plugin is starting results in an error. There is no check for when the plugin is stopping and so you should ensure you do not attempt to configure an instance if there is a chance the plugin is stopping. (Bug #27545850)
- When rejoining an instance to a cluster the displayed output has been improved. (Bug #27437389)
- The `dba.createCluster()` function fails if used on an instance with `innodb_page_size=4k`, because the `instance_name` column of the instances table in the InnoDB Cluster metadata

uses a `VARCHAR` of size 256, which is too large for the `innodb_page_size=4k`. However, the size of this column cannot be changed because it is used to hold hostnames of the cluster members which can have a length up to 255 bytes. This limitation is now validated when using `dba.createCluster()`, `dba.checkInstanceConfiguration()`, `dba.configureInstance()`, and `dba.configureLocalInstance()`. (Bug #27329079)

- The AdminAPI commands which manage sandboxes failed with an error when using a sandbox directory with non-ASCII characters in the path. This occurred when the user name had a non-ASCII character because the default sandbox directory is a subdirectory of the `$HOME` or `%userprofile%` path, which is usually based on the current user name. The fix adds Unicode support to the internal provision tool used by AdminAPI. (Bug #27181177)

- MySQL Shell now sets the value of `group_replication_local_address` based on `port` by calculating the result of `((port * 10) + 1)`. The `port` variable defaults to 3306, in which case MySQL Shell sets `group_replication_local_address` to `((3306 * 10) +1)`, resulting in port 33061 instead of the previous default of 13306. (Bug #27146799)

- The online help for all AdminAPI commands which require a connection displayed detailed information on connection data, which cluttered up the details of each command. Now, the help displays an overview of connection details and information about where else to get more help. (Bug #27146290)

- When creating a cluster on a Debian type platform with the default system configuration, the` `cluster.addInstance()` command failed with an error if the instance hostname was used in the connection parameter. This happened because on these platforms the hostname is resolved to the IP address 127.0.1.1 by default, but this IP address is not supported by the Group Replication Group Communication Service (GCS). The fix adds a validation to the `dba.createCluster()` and `cluster.addInstance()` functions to verify if the connection hostname resolves to 127.0.1.1 and issue an error in that case. (Bug #27095984)

- The `Cluster.forceQuorumUsingPartitionOf()` operation was not working with a non-root user, even if that user had all the required privileges. The fix ensures that the specified user is used to connect to all instances, instead of the root user. (Bug #27089930)

- Issuing `dba.createCluster()` in interactive mode as a user which did not have all the required privileges resulted in an unexpected halt. The error message generated when issuing `dba.configureLocalInstance()` and `dba.checkInstanceConfiguration()` if the account being used did not have the required privileges has also been improved. (Bug #27076753, Bug #27324699)

- When using `adoptFromGR` to convert a Group Replication group into an InnoDB Cluster, the output recommended adding instances. This message has now been improved to show that the instances were already added. (Bug #27061615)

- In interactive mode the `cluster.removeInstance()` AdminAPI function was not accepting the second parameter with the options dictionary. This prevented use of the `force` option, which failed with an error. (Bug #26986141)

  References: See also: Bug #27572618.

- The `cluster.addInstance()` function was not checking the validity of the `server_uuid` being added to the cluster. Now, the `server_uuid` of an instance joining the cluster is checked to be unique, and if it is not an error is generated and the instance is prevented from joining the cluster. (Bug #26962715)

- When setting up a cluster a replication user is created to enable distributed recovery. If MySQL Shell logging was enabled, the `CREATE USER` statements were not being correctly added to the log. (Bug #26938488)

- The `dba.checkInstanceConfiguration()` and `dba.configureLocalInstance()` operations were not correctly reporting any errors related to incorrect configuration of the instance's `server_id`. Now, when an error is reported it is displayed in the list of variables not meeting the InnoDB Cluster requirements.

  In addition, the X Plugin load has been removed from the `my.cnf` created by AdminAPI for sandbox instances running MySQL version 8.0.11 and later because X Plugin is installed by default for those versions. (Bug #26836230)

- When using `dba.configureLocalInstance()` with the `clusterAdmin` option to create a user which can administer the cluster, the created account had too many privileges. (Bug #26737608)

- When you create a cluster and add instances to it internal users are created, which are required by Group Replication for distributed recovery when instances join the cluster. These automatically generated replication users were not being removed from instances after removing them from the cluster or after dissolving the cluster. (Bug #26395608)

- If an instance contained InnoDB Cluster metadata but was stand alone, in other words it did not belong to a cluster, it was not possible to use `dba.dropMetadataSchema()`. (Bug #26315635)

- If you issued `dba.configureLocalInstance()` against an instance which was already valid for InnoDB Cluster usage and specified the `myCnf` option, but the `my.cnf` file was not readable, the operation would print that there are issues that need to be fixed. Now when you set the `myCnf` option, using either `dba.configureInstance()` and `dba.configureLocalInstance()`, the operations only use it if necessary. In other words, if the target instance is not valid for InnoDB Cluster usage and settings must be changed. (Bug #25702994)

- The `cluster.describe()` and `cluster.status()` AdminAPI methods return JSON objects containing the same information but some fields were identified by different tags. To make the tags consistent, the object returned from `cluster.describe()` has changed so that `instances` has been replaced with `topology`, and `host` with `address`. (Bug #25247515)

- Creating a cluster from an existing Group Replication group by using the `adoptFromGR:true` option on an instance which already belonged to cluster was not being correctly detected. Now such a situation is detected and generates an error. (Bug #25061891, Bug #25664766)

- MySQL Shell is able to create a session to an IPv6 address, but it failed to create an InnoDB Cluster using such connections, reporting an URI-related error. This was related to the encoding of the URI containing an IPv6 address, which has been fixed. The core issue, IPv6 support for AdminAPI is a known issue as Group replication requires an IPv4 network, see Group Replication Requirements. The implementation of InnoDB Cluster related operations has been modified in order to check if operations are executed using an IPv4 based session and IPv4 connection data. An exception is generated if these conditions are not met. (Bug #25042407)

- Attempting to add two different instances with the same label to an InnoDB Cluster correctly resulted in an error, however the instance with the duplicated label was being left with a partially initialized configuration. (Bug #24761416)

Functionality Added or Changed

- **Important Change; Microsoft Windows:** Before installing MySQL Shell, make sure you have the Visual C++ Redistributable for Visual Studio 2015 (available at the Microsoft Download Center) installed on your Windows system. This now applies for both Community and Commercial versions of MySQL Shell.

- **Important Change; X DevAPI:** The way which document IDs are generated has been changed. Now document IDs are generated by the server, rather than by the client. As a result the

`getLastDocumentID()`, `getDocumentId` and `getDocumentIDs()` methods have been removed. To get a list of the document IDs automatically generated by an 8.0.11 and later server, use `Result.getGeneratedIDs()`. The type of column generated for the document ID in a collection has changed from `VARCHAR(32)` to `VARBINARY(32)`. The generated document ID can be overridden manually by including an ID but you must respect the server generated IDs to avoid conflicts. If you are using InnoDB Cluster, use the `mysqlx_document_id_unique_prefix` variable to ensure your documents can be moved between replicasets.

Now, if you are adding documents to a collection on a server running a version of MySQL earlier than 8.0.11, you must manually include a document ID as these versions do not add the ID automatically.

- **X DevAPI:** Support for the `NOWAIT` and `SKIP LOCKED` `InnoDB` locking modes has been added to the lock operations. Now you can use these locking modes with the `lockShared()` and `lockExclusive()` methods, for example:

  - `Table.select().lockShared([LockContention])`
  - `Table.select().lockExclusive([LockContention])`
  - `Collection.find().lockExclusive([LockContention])`
  - `Collection.find().lockExclusive([LockContention])`

  where `LockContention` can be one of:

  - `DEFAULT` - if the function encounters a row lock it waits until there is no lock
  - `NOWAIT` - if the function encounters a row lock it aborts and generates an `ER_LOCK_NOWAIT` error
  - `SKIP_LOCKED` - if the function encounters a row lock it skips the row and continues

  For more information see Locking Read Concurrency with NOWAIT and SKIP LOCKED.

- The indexing of collections has been improved to make large collections of documents more efficient to navigate. Now, you can create an index based on one or more fields found in the documents in the collection, using a JSON document which maps fields from the collection's documents to MySQL types. The majority of MySQL types are supported, in addition to spatial indexes and GeoJSON data.

- MySQL Shell can now connect to a MySQL server with an account that uses the `caching_sha2_password` authentication plugin. Assuming the server is configured for encrypted connections, you can use such accounts over both X Protocol and the classic MySQL protocol. See Using Encrypted Connections.

  

  Important

  If you are not using encrypted connections, to connect over X Protocol with an account that uses the `caching_sha2_password` authentication plugin, the user's password must be stored in the cache. Currently there is no way to store the password over X Protocol if not using encrypted connections.

  When using the classic MySQL protocol for connections with such an account and unencrypted connections, you can configure MySQL for password exchange using an RSA key pair. MySQL Shell supports such connections and the following command options have been added:

  - Use the `--server-public-key-path` option to specify the RSA public key file.
  - Use the `--get-server-public-key` option to request the public key from the server.

For more information, see Caching SHA-2 Pluggable Authentication.

- MySQL Shell now has a configuration file which stores configuration changes across sessions. Use the new `\option` MySQL Shell command for querying and changing configuration options. Alternatively use the following methods with the `shell.options` object:

```
shell.options.set_persist(optionName, value)
shell.options.unset_persist(optionName, value)
```

  In addition the new `defaultMode` option has been added, which enables you to configure the programming language which MySQL Shell starts up in. You can override the default mode using the command options.

- The `util.checkForServerUpgrade([uri])` operation has been extended to check for the following incompatible features:
  - obsolete `sql_mode`
  - partitioned tables in shared tablespaces
  - removed functions

Bugs Fixed

- **X DevAPI:** The handling of SQL wildcard characters was corrected for the X DevAPI `Schema.getTable()`, `Schema.getCollection()` and `Session.getSchema()` functions. (Bug #26392984)
- When the `\quit` command was used to exit MySQL Shell, this event could be noted in the error log as an aborted connection. The issue has now been fixed. (Bug #27821045, Bug #90281)
- When the MySQL Shell command-line option `--json=raw` was used, the output was actually provided in pretty-printed format, and an empty string was displayed in place of error messages. These issues have now been corrected. (Bug #27733996, Bug #26737357)
- When MySQL Shell commands were executed from a script, interactive prompting for passwords and confirmations was not available. Now, interactive prompting is enabled by default when the commands are used in a script, as it is when they are used on the command line. The `--no-wizard` command line option disables interactive prompting for MySQL Shell commands used in both ways. (Bug #27702250)
- The `util.checkForServerUpgrade()` operation now prompts the user for a password interactively if the required password is not provided along with the command. If the `--no-wizard` option has been used to disable the connection wizard, missing credentials instead result in an error and the function is not executed. (Bug #27514395)
- The `util.checkForServerUpgrade()` operation was requiring the wrong privileges for the user passed to the function. The user now requires `ALL` privileges, and does not require the `GRANT OPTION` privilege. (Bug #27506702)
- The `util.checkForServerUpgrade()` operation was rejecting host names that included the % (percent) symbol or were specified as a numeric IP address. (Bug #27506079, Bug #27513260)
- By default, MySQL Shell connections are assumed to require a password, which is requested at the login prompt. A new MySQL Shell command-line option `--no-password` is provided to explicitly specify that no password is used, and to disable the password prompt. The `--no-password` option can be used if socket peer-credential authentication is in use (for Unix socket connections), or for any

authentication method where the user has a password-less account (though note that this situation is insecure and not recommended).

The methods previously provided by MySQL Shell for specifying that no password is used for the connection are still valid, and can be used instead of the `--no-password` option. These methods are as follows:

- If you are connecting using a URI type string, place a `:` after the user name in the URI type string but do not specify a password after it.
- If you are connecting using individual parameters, specify the `--password=` option with an empty value.

(Bug #26986360)

- When an error occurred while returning a result set, the fetch operation was interrupted but the associated error was not reported. The error is now reported correctly. (Bug #26906527)
- Support for microseconds has been added to the `mysqlx.dateValue()` function and the `Date` object. (Bug #26429497)
- Argument validation and error messages were improved for the `mysqlx.dateValue()` function. (Bug #26429426, Bug #26429377)
- The `db` global object is not available for connections in SQL mode. This reference has been removed from the message returned by the `\connect` command in that situation. (Bug #26428665)
- A shortcut to uninstall MySQL Shell is no longer provided in Microsoft Windows menus in the group of installed MySQL programs. Uninstallation of MySQL Shell should be handled through MySQL Installer. (Bug #26317449)
- The runtime timer for MySQL Shell, which reports the time taken for each query execution, has been refactored to provide increased precision of 4 digits for fractional seconds. (Bug #25976636, Bug #86135)
- In the event of a disconnection, MySQL Shell did not reconnect to the schema that was in use before the connection was lost. The last active schema set by the user is now restored during the automatic reconnection process, and during a manually triggered reconnection using the `\reconnect` command. (Bug #25974003, Bug #86115)
- MySQL Shell no longer attempts to reconnect automatically when the connection to the server is lost. A new MySQL Shell command `\reconnect` is provided, which makes MySQL Shell try several reconnection attempts for the current global session with the existing connection parameters. If those attempts are unsuccessful, you can make a fresh connection using the `\connect` command and specifying the connection parameters. (Bug #25105307)
- A memory leak was fixed that occurred if MySQL Shell was started and a connection was made to the MySQL server, then the user exited MySQL Shell without executing any commands. (Bug #24794589)

Changes in MySQL Shell 8.0.5 - 8.0.10 (Skipped version numbers)

There are no release notes for these skipped version numbers.

Changes in MySQL Shell 8.0.4 (2018-01-25, Release Candidate)

- AdminAPI Added or Changed Functionality
- AdminAPI Bugs Fixed

- Functionality Added or Changed
- Bugs Fixed

AdminAPI Added or Changed Functionality

- When you create clusters or add instances you can now override the default group name, local addresses, and group seeds. This makes it easier to customize your clusters. The following options were added to the `dba.createCluster()` and `cluster.addInstance()` commands:

  - use `groupName` with `dba.createCluster()` to set the name of the cluster
  - use `localAddress` to set the address which an instance provides to communicate with other instances
  - use `groupSeeds` to set the instances used as seeds when instances join the cluster

  For more information, see Customizing InnoDB Clusters. (Bug #26485254, Bug #26838005)

- Connections to an InnoDB cluster have been simplified. Now, when you issue `dba.getCluster()` and the active Shell session is not a connection to the primary, the cluster is queried for the primary information and a connection to the primary is automatically opened. This ensures that configuration changes can be written to the metadata. As part of this improvement you can now configure the type of MySQL Shell connection to an InnoDB Cluster with the following options which have been added:

  - `--redirect-primary`
  - `--redirect-secondary`
  - `--cluster`

  Additionally the following changes were made:

  - The `dba.resetSession()` method has been removed.
  - A new `disconnect()` method has been added to the `cluster` object, which closes all internal sessions opened by the cluster. InnoDB Cluster operations on a disconnected cluster object result in an error.
  - The output of the `Cluster.status()` method now includes the `groupInformationSourceMember` field, which shows the URI of the internal connection used by the cluster object to obtain information about the cluster.

  References: See also: Bug #25091586.

AdminAPI Bugs Fixed

- Account validation did not work correctly unless the session account existed. Now, validation is done using the account that was authenticated by the server. (Bug #26979375)
- The AdminAPI in MySQL Shell for working with InnoDB cluster only supports TCP connections to server instances. The AdminAPI now checks that a TCP connection is in use before starting an operation that requires database access, instead of attempting the operation with another connection type and not succeeding. (Bug #26970629)
- If you issued `STOP GROUP REPLICATION` on an instance that belonged to a cluster, attempting to rejoin the instance to the cluster failed because the wrong Group Replication seeds were being used.

Now, `Cluster.rejoinInstance()` correctly sets `group_replication_group_seeds` based on the `group_replication_local_address` of all currently active instances in the cluster. (Bug #26861636)

- Sometimes the `dba.addInstance()` command failed with an error indicating that the server was in `RECOVERING` state despite being `ONLINE`. The fix ensures the correct instance state is returned. (Bug #26834542)
- If the user running MySQL Shell did not have write permissions to the option file configured by AdminAPI, no error was displayed. (Bug #26830224)
- With the addition of WL#10470, the default value of `server_id` has changed to 1. As the server ID has to be unique for each instance, this caused issues with AdminAPI. Now, server instances with `server_id=1` are correctly identified as incorrectly configured for InnoDB Cluster use. (Bug #26818744)
- AdminAPI now supports Python 2.6 in addition to Python 2.7, removing the need to manually install on Oracle Linux 6. (Bug #26809748)
- After removing an instance from a cluster using `Cluster.removeInstance()`, the instance silently rejoined the Group Replication group after it restarted. This happened because `group_replication_start_on_boot` was set to `ON` by default. Now, for instances running MySQL version 8.0.4 and later, the fix sets `group_replication_start_on_boot` to `OFF` in the option file. For instances running a MySQL version earlier than 8.0.4, a warning is issued to tell you to manually edit `group_replication_start_on_boot` in the instance's option file to avoid the issue. (Bug #26796118)
- Using AdminAPI commands on Windows that required SSL resulted in an error due to the Python version being used. (Bug #26636911)
- Creating an InnoDB Cluster from an existing Group Replication deployment, by using the `adoptFromGR` option with the `dba.createCluster()` command, would fail with an error stating that the instance was already part of a replication group. The issue was only present in the MySQL Shell default wizard mode. The fix ensures that the interactive layer of the `dba.createCluster()` command allows the use of the `adoptFromGR` option. (Bug #26485316)
- The warnings generated when creating and adding sandbox instances have been improved. (Bug #26393614)
- When working with instances that had `require_secure_transport=ON`, AdminAPI commands that required a connection to the instance failed. (Bug #26248116)
- The `Cluster.dissolve()` command was trying to stop Group Replication on all of the instances registered in the metadata which lead to connection errors if any of those instances were not contactable, in other words with the state `(MISSING)`. The fix ensures that only instances which can be contacted, in other words with the state `ONLINE`, are stopped. (Bug #26001653)
- When adding instances to an InnoDB Cluster using the appropriate AdminAPI operations, checks are performed to verify the compatibility of any existing tables. If incompatible tables (for example using `MyISAM`) are detected then an error is issued. However the error message was referring to an option not available for the AdminAPI operations: `--allow-non-compatible-tables`. (Bug #25966731)
- The `cluster.rejoinInstance()` command attempted to rejoin an instance even if was already part of the cluster. Now, only instances in the `MISSING` state are accepted by `cluster.rejoinInstance()`. Attempting to rejoin an instance in any other state fails with an error. (Bug #87873, Bug #26870329)
- On Unix, if Python 3 was installed AdminAPI commands failed. (Bug #87731, Bug #26785584)

- When using the `dba.checkInstanceConfiguration()` and `dba.configurelocalinstance()` commands, the account being used was not being checked if it had enough privileges to actually execute the command. The fix ensures that account has the required privileges before proceeding. This also required a change of the privileges given to `clusterAdmin` users. (Bug #87300, Bug #26609909)

Functionality Added or Changed

- **X DevAPI:** A new `patch()` operation has been added to the `modify()` function which enables you to modify a document by merging in a set of changes. For more information see `JSON_MERGE_PATCH()`.
- **X DevAPI:** MySQL Shell performs automatic `_id` generation on collection add operations when no `_id` is specified on the documents being added. The autogenerated _id is created using the UUID() function. Now, the order of the tokens used has changed from `aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee`, where `eeeeeeeeeeee` is the MAC Address, to `eeeeeeeeeeee-dddd-cccc-bbbb-aaaaaaaa`.
- **X DevAPI:** Sessions created by X DevAPI using either `mysql.getClassicSession(connection_data)` or `mysqlx.getSession(connection_data)` now use `ssl-mode=REQUIRED` as the default if no `ssl-mode` is provided, and neither `ssl-ca` nor `ssl-capath` is provided. If no `ssl-mode` is provided and any of `ssl-ca` or `ssl`-capath is provided, all Sessions created by MySQL Shell now default to `ssl-mode=VERIFY_CA`.
- **X DevAPI:** In addition to the existing CRUD commands which work on documents in a collection by matching a filter, the following operations have been added to enable you to work with single documents:
  - `Collection.replaceOne(string id, Document doc)` updates (or replaces) the document identified by `id` with the provided one, if it exists.
  - `Collection.addOrReplaceOne(string id, Document doc)` add the given document. If the `id` or any other field that has a unique index on it already exists in the collection, the operations updates the matching document instead.
  - `Document Collection.getOne(string id)` returns the document with the given `id`. This is a shortcut for `Collection.find("_id = :id").bind("id", id).execute().fetchOne()`.
  - `Result Collection.removeOne(string id)` removes the document with the given `id`. This is a shortcut for `Collection.remove("_id = :id").bind("id", id).execute().`

  Using these operations you can reference documents by ID, making operations on single documents simpler by following a load, modify and save pattern such as:

```
doc = collection.getOne(id);
doc["address"] = "123 Long Street";
collection.replaceOne(id, doc);
```

- **X DevAPI:** You can now use savepoints with MySQL Shell sessions. The following methods have been added to the `Session` object:
  - Use `setSavepoint()` to generate a savepoint. The server executes the `SAVEPOINT` SQL statement and returns the generated savepoint name.
  - Use `setSavepoint(name)` to specify the name used by the `SAVEPOINT` SQL statement.
  - Use `releaseSavepoint(name)` to execute the `RELEASE` SQL statement.
  - Use `rollbackTo(name)` to execute the `ROLLBACK TO name` SQL statement.

  Any names passed to these functions are checked to make sure that the name is not null or an empty string. Names such as `''`, `""`, ``` `` ```, and so on are not allowed even though they are allowed

by the server. For more information, see SAVEPOINT, ROLLBACK TO SAVEPOINT, and RELEASE SAVEPOINT Statements.

- **X DevAPI:** X DevAPI now supports row-locking for CRUD operations on tables and collections. Use the `lockShared()` method for write locks and the `lockExclusive()` method for read locks, which have been added to `find()` and `select()`. Either method can be called any number of times, including none, in any combination. But the locking type to be used is always the last one called. Once you call a lock method you can only then execute the operation, calling `execute()` or `bind()`. For more information, see InnoDB Locking.

- **X DevAPI:** The X DevAPI drop operations have been improved. Now, the `drop()` methods are at the same level as the `create()` methods and they all return nothing. There is now no need to use `execute()` after the drop method. If a drop method is called on an object which no longer exists in the database there is now no error.

  This modifies `Session` objects so that:

  - `dropSchema()` returns Nothing
  - `dropTable(String schema, String name)` is removed
  - `dropCollection(String schema, String name)` is removed
  - `dropView(String schema, String name)` is removed

  This modifies Schema objects so that:

  - Nothing `dropCollection(String name)` is added

  This modifies Collection objects so that:

  - `dropIndex(String name)` is a direct operation, meaning `.execute()` is no longer needed
  - `dropIndex(String name)` returns Nothing

- **X DevAPI:** The `mysql` module, used with classic sessions for SQL connections to instances, has been streamlined. This has resulted in the following changes:

  - The new `getSession()` function has been added to the `mysql` module to create a classic session. This function works in the same way as the existing `mysql.getClassicSession()` function.
  - The `runSql()` method of `ClassicSession` has been extended to take a list of arguments to replace with placeholders in the query. For example, now you can issue:

    ```
    session.runSql("select * from tbl where name = ? and age > ?", ["Joe", "20"])
    ```

    Additionally a `query()` function has been added to `ClassicSession` as an alias to `runSql()`, making it consistent with `Session.query()`.

  - The following functions have been removed from `ClassicSession`:
    - `createSchema()`
    - `getSchema()`
    - `getDefaultSchema()`
    - `getCurrentSchema()`

    - `setCurrentSchema()`
    - `ClassicSchema()`
  - The `ClassicTable` object has been removed.
  - The following `Table` and View DDL functions have been removed from `Schema` objects:
    - `dropTable()`
    - `dropView()`
  - The behavior of the global `db` variable has been modified. When a global connection is created and the connection data includes a default schema, the global `db` variable is set to the corresponding `Schema` object. Now, this is not done for connections through the classic MySQL protocol, such as those created by `ClassicSession`.
- MySQL Shell now supports autocompletion of text preceeding the cursor by pressing the Tab key. Autocompletion is available for:
  - Built-in MySQL Shell commands, for example typing **`\con`** followed by the Tab key completes to `\connect`.
  - SQL, JavaScript and Python language keywords depending on the current MySQL Shell mode.
  - Table names, column names, and active schema names in SQL mode, based on the current default schema.

  Autocompletion can be configured using the following command options:
  - `--name-cache`
  - `--no-name-cache`
- You can now use Unix sockets for X Protocol connections. Socket file paths in URI type strings should be either percent encoded, such as `root@/home%2Fuser%2Fmysql-sandboxes%2F3310%2Fsandboxdata%2Fmysqlx.sock`, or surrounded by parenthesis such as `root@(/home/user/mysql-sandboxes/3310/sandboxdata/mysqlx.sock)`. The `--socket` option cannot be combined with the `--port` option. See Connecting to the Server Using URI-Like Strings or Key-Value Pairs.
- Use `util.checkForServerUpgrade([uri])` to check if a server can be upgraded from the version it is currently running to the next major series release. For example, you can verify that a server instance running MySQL 5.7 satisfies the upgrade prerequisites for MySQL 8.0, see Preparing Your Installation for Upgrade. To verify a server instance at a URI type string such as `user@example.com:3306` issue:

  ```
  util.checkForServerUpgrade(['user@example.com:3306'])
  ```

  MySQL Shell provides a report on anything in the table space which would cause a problem when upgrading the instance to MySQL 8.0.
- MySQL Shell builds against MySQL server version 8.0.4 and OpenSSL.
- MySQL 8.0.4 uses the `caching_sha2_password` authentication plugin and encrypted connections by default. This means any new accounts created in MySQL use these features. The new authentication method is completely transparent if the connection is made with encrypted connections enabled, which

is the default, and the preferred mode of connection. If encrypted connections are explicitly disabled, the following limitations apply:

- X Protocol sessions require encrypted connections.
- If the caching_sha2_password plugin reports an error such as `Authentication requires a secure connection`, it is not possible to connect to the account with MySQL Shell, until either an encrypted connection is made or the `mysql` client is used to clear the error.

See Using Encrypted Connections and Caching SHA-2 Pluggable Authentication.

Bugs Fixed

- Attempting a server connection without specifying the port or socket, when using an expired or temporary password, failed with an error requiring a password reset. The default connection data is now set at an appropriate point in connection processing, so the connection succeeds. (Bug #27266648)
- MySQL Shell required the option name `ssl-ciphers` instead of the standard MySQL option name `--ssl-cipher` to specify the list of permitted ciphers for connection encryption. The standard option name `--ssl-cipher` is now required in MySQL Shell. (Bug #27185275)
- The `--show-warnings` option was not working in MySQL Shell. (Bug #27036716)
- The command line options `--classic`, `--node`, `--sqln`, and `--ssl` were stated as deprecated, but actually had been removed. MySQL Shell now handles the options again, but prints a warning message when they are used. The deprecated options are processed as their replacement options, as follows:
  - `--classic` is processed as `--mysql`
  - `--node` is processed as `--mysqlx`
  - `--sqln` is processed as `--sqlx`
  - `--ssl` is processed as `--ssl-mode`

  (Bug #27012385)
- The output of `\status` when using a Unix socket for connections was showing a relative path. Now the absolute path of the socket is shown. (Bug #26983193)
- MySQL Shell now automatically pre-loads the built-in `mysql` and `mysqlx` API modules when it is invoked in batch mode, as well as when it is used in interactive mode. (Bug #26174373)
- The user configuration path for MySQL Shell defaults to `%AppData%\MySQL\mysqlsh\` on Windows, and `~/.mysqlsh/` on Unix. This directory is used for the MySQL Shell history file (`history`), log file (`mysqlsh.log`), and theme file (`prompt.json`). It is also the final location that MySQL Shell searches for startup scripts (`mysqlshrc.js` or `mysqlshrc.py`).

  You can now override the user configuration path by specifying an alternative path using the `MYSQLSH_USER_CONFIG_HOME` environment variable. A directory specified by that environment variable is used by MySQL Shell for the user configuration data in place of `%AppData%\MySQL\mysqlsh\` on Windows, and `~/.mysqlsh/` on Unix. (Bug #26025157)
- For file processing, MySQL Shell now expands a leading tilde in a file path to the appropriate home directory. MySQL Shell identifies the home directory using a relevant environment variable, or looks it up for the logged-in user. (Bug #25676405)

- MySQL Shell now provides a `program_name` connection attribute to the server at connect time, with the value `mysqlsh`. Connection attributes are displayed in the Performance Schema connection attribute tables. (Bug #24735491, Bug #82771)
- Running `help()` in MySQL Shell in Python mode caused an AttributeError. (Bug #24554329, Bug #82767)
- Arrays and Objects now accept the `IN` operator. For example:

```
collection.find("'Fred' IN username")
```

Changes in MySQL Shell 8.0.3 (2017-09-29, Development Milestone)

MySQL Shell now synchronizes the first digit of its version number with the (highest) MySQL server version it supports. This change makes it easy and intuitive to decide which client version to use for which server version. MySQL Shell now uses the same version number as MySQL Server.

MySQL Shell 8.0.3 is the first release to use the new numbering. It is the successor to MySQL Shell 8.0.0.

- AdminAPI Added or Changed Functionality
- AdminAPI Bugs Fixed
- Functionality Added or Changed
- Bugs Fixed

AdminAPI Added or Changed Functionality

- With the addition of WL#10611 and WL#10960 , it was not possible to add or rejoin instances that belonged to a cluster (or a replication group) because `super_read_only=ON` was being set by Group Replication when stopping. To ensure that AdminAPI supports instances running MySQL 8.0.2 and later, the following functions have been modified:
  - `dba.configureLocalInstance()`
  - `dba.createCluster()`
  - `dba.rebootClusterFromCompleteOutage()`
  - `dba.dropMetadataSchema()`

  Now, if any of these functions is issued against an instance which has `super_read_only=ON`, in interactive mode you are given the choice to set `super_read_only=OFF`. To force the function to set `super_read_only=OFF` in a script, pass the `clearReadOnly` option set to `true`. For example `dba.configureLocalInstance({clearReadOnly: true}).` For more information see Super Read-only and Instances. (Bug #26422638)

AdminAPI Bugs Fixed

- When using the `clusterAdmin` option, the created account did not have all of the correct privileges. (Bug #26523629)
- When using the `multiMaster` option with `dba.createCluster()`, the warning displayed in interactive mode was not being logged. (Bug #26385634)
- When making cluster topology or membership changes, AdminAPI was not taking into consideration the value of `group_replication_group_name`, which could lead to incorrect, non-deterministic results in

scenarios such as a split brain. Now, the following commands validate the InnoDB cluster Metadata and the corresponding instance's `group_replication_group_name` value:

- `dba.getCluster()`
- `Cluster.rejoinInstance()`
- `Cluster.forceQuorumUsingPartitionOf()`

If the values of `group_replication_group_name` do not match, the commands abort with an error.

`dba.rebootClusterFromCompleteOutage()` was also updated to ensure that the `group_replication_group_name` variable has not been changed before rejoining the instance. (Bug #26159339)

- AdminAPI now always uses the active user value for the current `mysqlsh` session, whether the value was explicitly specified by the user or is the result of an implicit default used by `mysqlsh`. (Bug #26132527)
- The checks performed by the AdminAPI upon issuing `dba.rebootClusterFromCompleteOutage()` were more strict than those required by Group Replication. Now, the AdminAPI considers tables with a Primary Key Equivalent (such as a Non Null Unique Key) as compatible, matching the current requirement for Group Replication. (Bug #25974689)
- The randomly generated passwords used by internal users were not compatible with instances running the Password Validation plugin. (Bug #25714751)
- It is no longer possible to use the `adoptFromGr` option with the `multiMaster` option. When adopting an existing group to an InnoDB Cluster, the group is adopted based on whether it is running as multi-primary or single-primary. Therefore there is no use for the `multiMaster` option when adapting a group. (Bug #25664700)
- Issuing `configureLocalInstance()` when using a URI that contained a user without the correct privileges resulted in an incorrect new user being created. Now, if the user in `configureLocalInstance()` URI does not have enough privileges to grant all the necessary privileges for the new user chosen during the interactive wizard configuration the user is not created. (Bug #25614855)
- Issuing `Cluster.rescan()` resulted in non-determinstic behavior which could produce incorrect JSON output, showing an instance that was already part of the cluster as belonging to the `newlyDiscoveredInstances[]` list and to the `unavailableInstances`[] list. This also resulted in MySQL Shell prompting to add or remove the instance from the cluster. (Bug #25534693)
- AdminAPI functions now accept the standard connection parameters as used by `shell.connect`. New validations have been added for when `require_secure_transport` is `ON`, now it is not possible to create a cluster with `memberSslMode:DISABLED` or to add an instance with `require_secure_transport=ON` to a cluster where `memberSslMode:DISABLED`. (Bug #25532298)
- The parsing of account names, for example when passing the `clusterAdmin` option to `dba.configureLocalInstance()` has been improved. (Bug #25528695)
- The file permissions of option files created by AdminAPI did not match those of options files created by MySQL install. (Bug #25526248)
- Issuing `configureLocalInstance()` twice could fail. (Bug #25519190)

- When passing the `rejoinInstances[]` option to `dba.rebootClusterFromCompleteOutage()`, if no `rejoinInstances[]` option was specified then members were being incorrectly handled during the rebuild. Now, instances that are eligible to be added to the `rejoinInstances[]` list but that are specified in the `removeInstances[]` list are skipped by the interactive wizard that tries to automatically build a `rejoinInstances[]` list if one was not provided. This fix also ensures that both interactive and noninteractive use of MySQL Shell correctly verify the `rejoinInstances[]` list does not contain a unreachable instance. (Bug #25516390)

- The error messages issued when the SSL mode used by the cluster and the one specified when issuing `addInstance()` command do not match have been improved. (Bug #25495056)

- When creating a sandbox instance using the `dba.deploySandboxInstance()` function in MySQL Shell, pressing **Ctrl+C** at the prompt for a MySQL root password for the instance did not cancel the deployment. (Bug #25316811)

- Issuing `removeInstance`() on the last member of a cluster, and particularly the seed member, was resulting in a cluster that could not be dissolved. Now, issuing `removeInstance()` on the last member of a cluster results in an error, and you must use `dissolve()` on that instance to ensure the cluster is correctly dissolved. (Bug #25226130)

- The output of `cluster.status()` now includes the `ssl` parameter, which shows whether secure connections are required by the cluster or disabled. (Bug #25226117)

- Attempting to create a multi-primary cluster in interactive mode failed unless you passed in the `{force: true}` option. Now when you confirm that you understand the impact of using multi-primary mode the command correctly creates a multi-primary cluster. (Bug #25034951)

- The `removeInstance()` was not working on stopped instances and it was not possible to remove an unavailable instance from the cluster. The fix adds a new option `force` to the removeInstance() command to enable you to remove instances from the metadata that are permanently not available, avoiding obsolete data from being kept in the metadata of the cluster. In addition the error message provided when not using the force option has been improved and the online help for the `removeInstance()` was also updated accordingly. (Bug #24916064)

- The error messages generated by issuing `dba.deployLocalInstance()` against an unsuitable or incompatible instance have been improved. (Bug #24598272)

- The `dba.createCluster()`, `dba.getCluster()`, and `dba.rebootClusterFromCompleteOutage()` functions have been updated to validate the cluster name, using the following rules:

  - Name must start with a letter or the _ character
  - Name can only contain alphanumeric characters and the _ character
  - Cannot be longer than 40 characters
  - Cannot be empty

  The `Cluster.addInstance()` function has been updated to validate the label used on an instance in the cluster, using the following rules:

  - Label can only contain alphanumerics or the _ character
  - Cannot be longer than 256 characters
  - Cannot be empty

(Bug #24565242)

Functionality Added or Changed

- **X DevAPI:** The types of session available have been simplified. `XSession` and `NodeSession` have been consolidated into `Session`. This has caused the following changes:
  - The following command options have been deprecated: `--node`, `--sqln`, `--classic`
  - The following command options have been introduced to replace the deprecated ones: `-ma`, `--mysqlx` (`-mx`), `--mysql` (`-mc`), `--sqlx`
  - The `\connect` MySQL Shell command no longer supports the arguments `-c`, and `-n`. Now the `\connect` command supports the argument `--mysqlx`(`-mx`) for creating X Protocol connections, and `--mysql`(`-mc`) for creating classic MySQL protocol connections.
- MySQL Shell now handles user interrupts, such as SIGINT, correctly. For example on Linux pressing Control-C when MySQL Shell is not executing anything exits the application. In SQL mode, interruption sends a `KILL QUERY` statement to the active MySQL Shell session from a new temporary session, resulting in the server interrupting the query and returning an error (or in an early return with no error in some cases, like the `sleep()` function). In JavaScript or Python scripting modes, how interruption behaves depends on the specific function being executed. If what is being executed is language code (such as a while loop and other normal script code), an exception is generated in the active language, which causes the code to stop executing. The exception may be caught by the script, but if not, the execution control returns to MySQL Shell. (Bug #24757361)
- MySQL Shell now includes a history function that stores the code which you issue. The history can be saved, searched, and filtered. A new mechanism to customize the MySQL Shell prompt has been added. Information such as the current mode (SQL, JavaScript, or Python), session information (host, uri, port and so on), the current active schema and others can be included in the prompt through variables. The customization information is self-contained in JSON theme files, which can be shared between users. MySQL Shell now supports unicode if the terminal used to run MySQL Shell supports it. Similarly if the terminal supports color, MySQL Shell can be configured to use colors in the theme.
- The connection options passed to MySQL Shell, such as `sslMode` and so on, have been changed to use dashes and no longer be case sensitive. The options are now:
  - `sslMode` is now `ssl-mode`
  - `sslCa` is now `ssl-ca`
  - `sslCaPath` is now `ssl-capath`
  - `sslCert` is now `ssl-cert`
  - `sslKey` is now `ssl-key`
  - `sslCrl` is now `ssl-crl`
  - `sslCrlPath` is now `ssl-crlpath`
  - `sslCiphers` is now `ssl-ciphers`
  - `sslTlsVersion` is now `tls-version`
  - `authMethod` is now `auth-method`

- The interpretation of the `document_path` field in operations such as `modify()` has been changed. Now, when the `document_path` is not set, operations apply to the whole document. All operations always preserve a document's `_id` field.

Bugs Fixed

- **X DevAPI:** Unsigned data could be incorrectly read from the database. (Bug #24912358)
- In MySQL Shell, the `Schema.getCollectionAsTable()` function and the `select()` method could not be used in the same Python statement. (Bug #26552804)
- MySQL Shell returned some elements of DATE and DATETIME values incorrectly, including month values and fractional seconds. (Bug #26428636)
- The month was incorrectly incremented on insertion of a timestamp in a table using MySQL Shell. (Bug #26423177)
- For columns with the `ZEROFILL` attribute, `NULL` was also returned padded with zeroes. (Bug #26406214)
- The output of the MySQL Shell `\status` command was enhanced with additional information. (Bug #26403909)
- The MySQL Shell help for the `\connect` command indicated that a connection name could be used instead of a URI string, which was incorrect. (Bug #26392676)
- The MySQL Shell command `\use` did not attempt to reconnect if the connection to the global session was lost. (Bug #25974014, Bug #86118)
- The short form `-?` can now be used as an alias for the `--help` command-line option in MySQL Shell. (Bug #25813228)
- The MySQL Shell command history displayed the commands that were used to automatically import the `mysql` and `mysqlx` API modules when MySQL Shell started. (Bug #25739185)
- The MySQL Shell command history displayed the contents of scripts that were run using the `\source` MySQL Shell command. (Bug #25676495)
- The `mysqlx.getNodeSession()` function in MySQL Shell now returns an error if an unrecognized connection option is provided. (Bug #25552033)
- The `--ssl` option has been deprecated, use the `--ssl-mode` option. Now, if you use the `--ssl` option, a deprecation warning is generated and the `--ssl-mode` option is set to either `DISABLED` or `REQUIRED` based on the value used with the `--ssl` option. (Bug #25403945)
- MySQL Shell did not exit gracefully when the user did not have a valid and accessible home directory. (Bug #25298480)
- MySQL Shell created a logger but did not deallocate it on exiting the shell. (Bug #25238576)
- MySQL Shell could hang when **Ctrl+C** was used to exit the shell. (Bug #25180850, Bug #84022)
- The parsing of Unix sockets provided as part of a URI has been improved. (Bug #24905066)
- MySQL Shell now accepts Unicode characters as input. (Bug #23151666, Bug #81176)

Changes in MySQL Shell 8.0.2 (Skipped)

Version 8.0.2 has no release notes, or they have not been published because the product version has not been released.

Changes in MySQL Shell 8.0.1 (Skipped)

Version 8.0.1 has no release notes, or they have not been published because the product version has not been released.

Changes in MySQL Shell 8.0.0 (2017-07-14, Development Milestone)

AdminAPI Bugs Fixed

- Executing AdminAPI commands on a server with a version of Python lower than 2.7 was failing without the correct error message. (Bug #25975317)
- When using MySQL Shell on Windows any files created or opened, for example those used during `dba.createSandboxInstance()`, could not be deleted. (Bug #25789094)
- The help for `dba.configureLocalInstance(instance[, options])` has been improved to describe the returned JSON object. (Bug #25703028)
- When using `dba.deploySandboxInstance()` and passing in `sandboxDir`, the specified path must not exceed 89 characters. (Bug #25485035)
- `removeInstance()` resulted in unexpected behavior in some cases, for example when an empty password was passed as part of the URI to the instance. (Bug #25111911)
- Creating Classic sessions that connect using Unix sockets now uses the correct defaults such as hostname. This resolves the previous limitation of using Unix sockets to connect to InnoDB cluster instances. See MySQL Shell Connections for information on how the defaults are applied to socket connections. (Bug #24848763, Bug #26036466)

  References: See also: Bug #24911068.
- On an instance configured as a multithreaded replica, in other words `slave_parallel_workers` set to greater than 0, and with `slave_parallel_type=DATABASE`, `dba.checkInstanceConfiguration()` was not detecting that the instance was not correctly configured for InnoDB cluster usage.
- If `removeInstance()` failed due to a connection error, an error was reported but the instance was incorrectly removed from the InnoDB cluster metadata, and remained part of the replication group. The fix ensures the metadata is correctly updated according to the result of `removeInstance`().
- In a situation where a new primary instance was elected, adding a new instance to the cluster resulted in an error due to a failed connection to the previous primary instance.
- The functions that modify server variables, such as `dba.createCluster()` and `dba.validateInstance()` now provide more information in interactive mode output and log output about server variables which are changed when executed.
- Deploying instances to paths with directories that contained spaces was failing without error. Use double backslash to specify such paths, for example `D:\\Cluster\\foo bar`.

- The Cluster object obtained from functions such as `dba.createCluster()` or `dba.getCluster()` became unusable once the Shell session in which the object was created is was connected to a different server. The fix modifies the Cluster object so that:
  - The Cluster object holds an internal reference to the Session from which it was created or retrieved.
  - AdminAPI functions that modify the Cluster are made using the session referenced by the object.

Functionality Added or Changed

- Calling the `modify()` or `remove()` function without a parameter caused the function to be executed against the whole collection, which could cause unexpected results such as deleting all rows in a table. To avoid this and make the behavior consistent with `update()` and `delete()`, a client-side exception is now thrown if the `modify()` or `remove()` function is called without a parameter. Now, to execute the `modify()` or `remove()` function against a collection call them with an expression that evaluates to `true`, for example `remove('true')` or `modify('true')`.

Bugs Fixed

- The options in the MySQL Shell options dictionary are now fully documented. (Bug #25701345)
- `shell.connect()` did not report an error if an invalid argument was used. An `ArgumentError` is now issued for any invalid argument.

  The following mutually exclusive pairs of options are now checked, and an error is issued if both are specified:

  - `--password` and `--dbPassword`
  - `--user` and `--dbUser`
  - `--port` and `--socket`

  (Bug #25268670)

  References: See also: Bug #24911173.
- A number of issues with the output of `shell.help("prompt")` have been corrected. (Bug #25026855, Bug #25242638, Bug #25676343, Bug #25176769)
- MySQL Shell now displays an invalid year as `0000`, matching the behavior of the MySQL prompt, rather than as `0`. (Bug #24912061)
- MySQL Shell did not display fractional seconds for values in DATETIME columns. (Bug #24911885)
- Some issues with the MySQL Shell command line help output were fixed. (Bug #24841749, Bug #24841493, Bug #24910540)
- URIs were incorrectly parsed in MySQL Shell when passwords were hidden. (Bug #24793956)
- `mysqlsh` stopped responding if the `\source` command was given a directory (rather than file) argument. (Bug #23097932, Bug #81060)